

01/767231

APPENDIX

CELLULAR PHONE ACCOUNTING SYSTEM

DONALD S. MCGREGOR
GREGORY M. MCGREGOR

DKT 11557

BIELEN, PETERSON & LAMPE
1990 N. CALIFORNIA BLVD.
SUITE 720
WALNUT CREEK, CALIFORNIA 94596

(510) 937-1515


```

/*-----
windows.c
    A Very Simple windows package for windowing

```

Written By: Greg McGregor 1990

```

REVISED:      What was revised?
CMM 7-30-1991      Nothing

```

```

/*-----
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <alloc.h>
#include <process.h>
#include <time.h>
#include <\h2\malloc\galloc.h>

```

```

#define SINGLEFRAME 1
#define DOUBIEFRAME 2

```

```

#define CenterUpperTitle 1

```

```

#define Black      0
#define Blue       1
#define Green      2
#define Cyan       3
#define Red        4
#define Magenta    5
#define Brown      6
#define Lightgray  7
#define Darkgray   8
#define Lightblue  9
#define Lightgreen 10
#define Lightcyan  11
#define Lightred   12
#define Lightmagenta 13
#define Yellow     14
#define White      15
#define Blink      128

```

```

int UL,DU,UR,LR,FD,LL;

```

```

#define D_UL 201 /* double Line Attributes */
#define D_DU 186
#define D_UR 187
#define D_LR 188
#define D_FD 205
#define D_LL 200

```

```

#define S_UL 218
#define S_DU 179
#define S_UR 191
#define S_LR 217
#define S_FD 196
#define S_LL 192

```

```

int MAX_WINDOWS = 20;

```



```
#define TRUE 1
#define FALSE 0
```

```
/*
 * window structure
 */
typedef struct {
    int x1,y1,x2,y2;
    int foreground,background;
    int cursor_on,wrap_on,hidden,frame_on;
    int frame;
    int frame fore,frame back;
} windef;
```

```
int WINDOW_EXPLODE;
int win_delay;
```

```
/*
 * window type
 */
```

```
typedef struct {
    int window_no;
    void *save_bottom;
    int shaded; /* could the window be shaded */
    windef window_struct;
} wintype;
```

```
/*
 * window stack
 */
typedef struct {
    int window_no;
    int open;
} open windows struct;
```

```
/*
 * Window Call Stack
 */
typedef struct {
    int window_no;
    wintype w;
} window stack struct;
```

```
/*
 * Pick list type
 */
typedef struct {
    char list[10][30];
} pick_list_type;
```

```
window_stack_struct window_stack[21]; /* MAX_WINDOWS + 1 */
```

```
open_windows_struct window_pool[21]; /* pool of all available windows */
int current_window;
wintype current_top,windows_error_wt;
```



```

windef windows_error = {10,10,70,15,White,Red,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
                        White,Red};

```

```

void flatWindow (int x,int y,int x1,int y1,int back,int textC);
void init_windows (void);
wintype windowopen (windef *wd);
void use (wintype w);
int windowclose (wintype w);
void settitle (wintype win, char *title, int mode);
void close_all_windows (void);
void explode (int x,int y,int x1,int y1,int back,int textC);

```

```

int P init windows = FALSE;

```

```

/*-----
init_windows
-----*/
void init_windows ()
{
    int i;
    current_window = -1;
    for (i=1;i<=MAX_WINDOWS;i++) {
        window_pool[i].window_no = i;
        window_pool[i].open = FALSE;
        window_stack[i].window_no = -1;    /* clear stack */
    }
    WINDOW_EXPLODE = FALSE;
    win_delay = 44;
}

```

```

/*-----
get_video_memory : allocate memory for window
-----*/
char *get_video_memory (int x,int y,int x1, int y1)
{
    char *buff;
    size_t size;
    size = (2 * (x1 - x + 1) * (y1 - y + 1));
    buff = (char *) g_malloc (size);
    if (buff == NULL) {
        windows_error_msg ("get_video_memory : Can't Allocate memory");
    }
    return buff;
}

```

```

/*-----
push_stack : push window on stack
-----*/
void push_stack(wintype w)
{
    int i,ok,n;
    n = w.window_no;
    ok = FALSE;
    i = 1;
    while ( (i<=MAX_WINDOWS) && (!ok) ) {
        if (window_stack[i].window_no == -1) {
            ok = TRUE;
            window_stack[i].window_no = n;
            window_stack[i].w = w;
        }
    }
}

```



```

        }
        ++i;
    }
    if (lok) {
        windows error msg ("put stack: Stack overflow!");
    }
}

/*-----
pop_stack
-----*/
int pop_stack()
{
    int i,ok,val;
    i = MAX_WINDOWS;
    ok = FALSE;
    val = -1;
    while ( (i>=1) && (lok) ) {
        if (window_stack[i].window_no != -1) {
            val = window_stack[i].window_no;
            window_stack[i].window_no = -1;
            /* free video memory */
            g_free (window_stack[i].w.save_bottom);
            ok = TRUE;
        }
        --i;
    }
    if (lok) {
        windows error msg ("pop stack: Stack Underflow!");
    }
    return val;
}

```

```

/*-----
peek_stack
-----*/
int peek_stack ()
{
    int i;
    i = MAX_WINDOWS;
    while ( (i>=1) ) {
        if (window_stack[i].window_no != -1)
            return window_stack[i].window_no;
        --i;
    }
    return -1;
}

```

```

/*-----
pull_out_a_window; : return top window
-----*/
wintype pull_out_a_window (int id)
{
    int i;
    if (id == -1) return;
    for (i=1;i<=MAX_WINDOWS;i++){
        if (window_stack[i].window_no == id)
            return window_stack[i].w;
    }
}

```



```

}

```

```

/*-----
to_top_of_stack(i); put i on top of calling stack
-----*/

```

```

void to_top_of_stack(int x)
{
    int i,j,stack_pos,ok;
    window_stack_struct temp;
    stack_pos = -1;
    for (i=1;i<=MAX_WINDOWS;i++)
        if (window_stack[i].window_no == x){
            stack_pos = i;
            temp = window_stack[i];
        }
    if (stack_pos == -1) {
        windows_error_msg ("to_top_of_stack : Stack ERROR!");
    }
    /* shift stack all down one */
    j = stack_pos;
    ok = FALSE;
    while ( (j<=MAX_WINDOWS-1) && (!ok) ) {
        if (window_stack[j+1].window_no == -1){
            ok = TRUE;
        } else {
            window_stack[j] = window_stack[j+1];
            ++j;
        }
    }
    /* move x to top of stack */
    window_stack[j] = temp;
    window_stack[j].window_no = x;
}

```

```

/*-----
set_single : set single line frame attrb
-----*/

```

```

void set_single ()
{
    UL = S_UL;
    DU = S_DU;
    UR = S_UR;
    LR = S_LR;
    FD = S_FD;
    LL = S_LL;
}

```

```

/*-----
set_double : set double line frame attrb
-----*/

```

```

void set_double ()
{
    UL = D_UL;
    DU = D_DU;
    UR = D_UR;
    LR = D_LR;
    FD = D_FD;
    LL = D_LL;
}

```



```

/*-----
windowopen : open a window
-----*/

wintype windowopen (windef *wd)
{
    wintype wt;
    int i, found, val, size;
    char *shade;
    char shade_attr = 0x07;

    if (!P_init_windows) {
        init_windows ();
        P_init_windows = TRUE;
    }

    found = FALSE;
    i = 1;
    while ( (i <= MAX_WINDOWS) && (!found) ) {
        if (!window_pool[i].open) {
            wt.window_no = window_pool[i].window_no;
            found = TRUE;
            window_pool[i].open = TRUE;
        }
        ++i;
    }
    if (!found) {
        printf ("\n No More available windows..");
        wt.window_no = -1;
        return wt;
    }
    wt.window_struct = *wd;
    wt.shaded = FALSE;
    if ( (wd->x2 < 80) && (wd->y2 < 24) ) {
        wt.save_bottom = get_video_memory (wd->x1, wd->y1, wd->x2+1, wd->y2+1);
        if (!gettext (wd->x1, wd->y1, wd->x2+1, wd->y2+1, wt.save_bottom) ) {
            windows_error_msg ("Can't Open Window!");
            wt.window_no = -1;
            return wt;
        }
        wt.shaded = TRUE;
    } else {
        wt.save_bottom = get_video_memory (wd->x1, wd->y1, wd->x2+1, wd->y2+1);
        if (!gettext (wd->x1, wd->y1, wd->x2, wd->y2, wt.save_bottom) ) {
            windows_error_msg ("Can't Open Window!");
            wt.window_no = -1;
            return wt;
        }
        wt.shaded = FALSE;
    }
}

/* shade if there is room to shade */
if (wt.shaded) {
    size = (2 * (wd->x2 - wd->x1 + 1) * (wd->y2 - wd->y1 + 1));
    shade = get_video_memory (wd->x1, wd->y1, wd->x2, wd->y2);
    if (!gettext (wd->x1+1, wd->y1+1, wd->x2+1, wd->y2+1, shade) ) {
        windows_error_msg ("windowopen: gettext : FAILED!");
        wt.window_no = -1;
        return wt;
    }
}

```



```

    }
    for (i=1;i<size;i = i + 2) {
        shade[i] = shade_attr;
    }
    if (!puttext (wd->x1+1,
                  wd->y1+1,          /* shading here */
                  wd->x2+1,
                  wd->y2+1,
                  shade) ) {
        windows_error_msg ("windowopen: shading error");
        wt.window_no = -1;
        return wt;
    }
    g_free (shade);
} /* end shading */

if (wd->frame == SINGLEFRAME)
    set_single();
if (wd->frame == DOUBLEFRAME)
    set_double();
flatWindow (wd->x1,wd->y1,wd->x2,wd->y2,wd->background,wd->foreground);
push_stack (wt);
return wt;
}

/*-----
use;
-----*/

void use (wintype w)
{
    int val;
    if (w.window_no == peek_stack() ) {
        w = pull_out_a_window (w.window_no);
        window (w.window_struct.x1+1,w.window_struct.y1+1,
                w.window_struct.x2-1,w.window_struct.y2-1);
        textcolor (w.window_struct.foreground);
        textbackground (w.window_struct.background);
        return; /* top window already in use But still reset window coords*/
    }
    if (!window_pool[w.window_no].open) {
        windows_error_msg ("use : Window Not OPEN!");
    }

    w = pull_out_a_window (w.window_no);

    window (w.window_struct.x1+1,w.window_struct.y1+1,
            w.window_struct.x2-1,w.window_struct.y2-1);
    textcolor (w.window_struct.foreground);
    textbackground (w.window_struct.background);
    to_top_of_stack (w.window_no);
}

/*-----
restore_coords : reset window dimensions
-----*/

void restore_coords (wintype w)
{
    window (w.window_struct.x1+1,w.window_struct.y1+1,
            w.window_struct.x2-1,w.window_struct.y2-1);
}

```



```

/*-----
windowclose ;
-----*/
int windowclose (wintype w)
{
    int i;
    if (w.window_no != peek_stack()) {
        return FALSE;
    }
    if (w.shaded) { /* close shade, too, if applicable */
        if (!puttext (w.window_struct.x1,
                     w.window_struct.y1,
                     w.window_struct.x2+1,
                     w.window_struct.y2+1,
                     w.savc_bottom) ) {
            windows_error_msg ("windowclose: error exiting window");
        }
    } else {
        if (!puttext (w.window_struct.x1,
                     w.window_struct.y1,
                     w.window_struct.x2,
                     w.window_struct.y2,
                     w.savc_bottom) ) {
            windows_error_msg ("windowclose: error exiting window");
        }
    }

    for (i=1; i<=MAX WINDOWS; i++) {
        if (window_pool[i].window_no == w.window_no) {
            window_pool[i].open = FALSE;
        }
    }
    i = pop_stack(); /* pop off last window off of call stack */
    /* use window under one just erased */

    if (peek_stack() == -1) { /* no open windows..reset */
        return FALSE;
    }

    restore_coords (pull_out_a_window (peek_stack()) );
    textcolor (window_stack[peek_stack()].w.window_struct.foreground);
    textbackground (window_stack[peek_stack()].w.window_struct.background);

    return TRUE;
}

```

```

/*-----
settitle: settitle switches to this window
-----*/
void settitle (wintype win, char *title, int mode)
{
    int width, len, pos;
    window (win.window_struct.x1,
            win.window_struct.y1,
            win.window_struct.x2,
            win.window_struct.y2);
    width = win.window_struct.x2 - win.window_struct.x1;
    len = strlen (title);
    pos = (width / 2) - (len / 2);
}

```



```

    gotoxy (pos,1);
    cprintf ("%s",title);
    window (win.window_struct.x1 +1,
            win.window_struct.y1 +1,
            win.window_struct.x2 - 1,
            win.window_struct.y2 - 1);
}

```

```

/*-----
draw_lines-----*/

```

```

void draw_lines (x,y,x1,y1,back,textC)
int x,y,x1,y1,textC;
{
    int i,j;

    textcolor (textC);
    textbackground (back);
    window (x,y,x1,y1);
    clrscr ();

    cprintf ("%c",UL);
    for (i=x+1;i<x1;++i)
        cprintf ("%c",FD);
    cprintf ("%c",UR);

    for (j=y+1;j<y1;++j){
        gotoxy (1,j+1-y);
        cprintf ("%c",DU);
        gotoxy (x1+1-x,j+1-y);
        cprintf ("%c\n",DU);
    }
    gotoxy (1,y1-y);
    cprintf ("%c",LL);
    for (i=x+1;i<x1;++i)
        cprintf ("%c",FD);
    window (x1,y1,x1,y1);
    cprintf ("%c",LR);

    window (x+1,y+1,x1-1,y1-1);
}

```

```

/*-----
explode a window-----*/

```

```

void explode (x,y,x1,y1,back,textC)
int x,y,x1,y1,back,textC;
{
    int i,xx,yy,xx1,yy1;
    delay (0); /* calibrate clock */
    xx = (x+x1)/2-3;
    yy = (y+y1)/2-3;
    xx1 = xx + 6;
    yy1 = yy + 6;

    for (i=1;i<=3;++i) {
        draw_lines (xx,yy,xx1,yy1,back,textC);
        xx = (xx+x)/2;
        yy = (yy+y)/2;
        xx1 = (xx1+x1)/2;
    }
}

```



```

yy1=(yy1+y1)/2;
delay(win_delay);
}
}

```

```

/*-----
flatWindow
-----*/

```

```

void flatWindow (x,y,x1,y1,back,textC)
int x,y,x1,y1,back,textC;

```

```

{
int i,j;

```

```

if (WINDOW_EXPLODE) explode (x,y,x1,y1,back,textC);

```

```

textcolor (textC);
textbackground (back);
window (x,y,x1,y1);
clrscr ();

```

```

printf ("%c",UL);
for (i=x+1;i<x1;++i)
    printf ("%c",FD);
printf ("%c",UR);

```

```

for (j=y+1;j<y1;++j){
    gotoxy (1,j+1-y);
    printf ("%c",DU);
    gotoxy (x1+1-x,j+1-y);
    printf ("%c\n",DU);
}

```

```

gotoxy (1,y1+1-y);
printf ("%c",LL);
for (i=x+1;i<x1;++i)
    printf ("%c",FD);
window (x1,y1,x1,y1);
printf ("%c",LR);

```

```

window (x+1,y+1,x1-1,y1-1);
clrscr ();
}

```

```

/*-----
close_all_windows : close all windows free up memory
-----*/

```

```

void close_all_windows ()

```

```

{
int id;
wintype win;

```

```

    id = peek_stack ();
    while (id != -1) {
        win = pull_out_a_window (id);
        windowclose (win);
        id = peek_stack ();
    }
}

```

```

/*-----
ruff_area : clrscr with a ruff area image

```



```

-----*/
ruff_area (int x,int y,int x1,int y1,int t_color,int b_color)
{
int i,j;
char *sto; /* memory storage for shade */
    sto = get_video_memory (x,y,x1,y1);
    i = (2 * (x1 - x + 1) * (y1 - y + 1)); /* get size of area */

get_video_memory */
    for (j=0;j<i;j += 2) {
        sto[j] = 176; /* ruff char */
        sto[j+1] = (char)(t_color | b_color); /* fore, back color */
    }
    if (!puttext (x,y,x1,y1,sto)) {
        windows_error_msg ("ruff_area: puttext error!");
    }
    free (sto); /* free up memory */
}

/*-----
windows error msg
-----*/
windows_error_msg (char *s)
{
    windows_error_wt = windowopen (&windows_error);
    settitle (windows_error_wt,"Windowing Error!",CenterUpperTitle);
    clrscr ();
    cprintf ("%s",s);
    getch ();
    exit (0);
}

/*
//
// display_pick_list
//
*/
void display_pick_list (wintype wt,int item,pick_list_type *list,int items) {
int i;
    use (wt);
    clrscr ();
    for (i=0;i<items;i++) {
        if ( (i+1) == item) {
            gotoxy (1,i+1);
            textbackground (Black);
            textcolor (White);
            cprintf ("%c%s",0x10,list->list[i]);
        } else {
            gotoxy (1,i+1);
            textbackground (Cyan);
            textcolor (White);
            cprintf (" %s",list->list[i]);
        }
    }
}

/*
//

```

/* same formula as


```

// get_kb_char - with a timeout of 1 minute
// return 0F in upper byte if extended key
*/
int get_kb_char () {
char ch;
int x;
time_t start,current;
    start = current = clock ();
    while ( (lkbhit ()) && ( (current-start)/CLK_TCK < 60) )
        current = clock ();
    if ( !lkbhit () ) return ( FALSE );
    ch = getch ();
    if (ch == 0x00) {
        x = 0x0F00;
        ch = getch ();
    } else x = 0x0000;
    return ( x | ch );
}

/*
//
// pick_list
//
*/
int pick_list (pick_list_type *list,int items,char *title) {
window pick_win = {1,1,80,24,White,Cyan,FAISF,FALSE,FALSE,TRUE,SINGLEFRAME,
                    White,Cyan};
wintype pick_wt;
int cols,i,j,key;
    _setcursortype (_NOCURS);
    cols = 0;
    for (i=0;i<items;i++) {
        j = strlen (list->list[i]);
        if (j > cols) cols = j;
    }
    cols += 4;
    if (cols < strlen (title)) cols = strlen (title) + 2; /* minimum width */
    pick_win.x1 = 40 - (cols/2); /* center picklist on screen */
    pick_win.x2 = 40 + (cols/2);
    pick_win.y1 = 12 - ((items+2)/2);
    pick_win.y2 = 12 + ((items+2)/2);
    pick_wt = windowopen (&pick_win);
    setttitle (pick_wt,title,CenterUpperTitle);
    key = 0;
    i = 1;
    while ( (key != 0x000D) && (key != 0x001B) ) { /* enter and esc */
        if ( key == 0x0F50 ) /* down arrow */
            if ( i == items ) { i = 1; } else ++i;
        if ( key == 0x0F48 ) /* up arrow */
            if ( i == 1 ) { i = items; } else --i;
        display_pick_list (pick_wt,i,list,items);
        key = get_kb_char ();
        if ( key == 0x0000 ) return ( 0x001B ); /* time out return ESC */
    }
    windowclose (pick_wt);
    _setcursortype (_NORMALCURSOR);
    if (key == 0x000D) return ( i );
    return ( 0x001B );
}

```



```
}
```

```
/*
```

```
//
```

```
// add_to_pick_list
```

```
//
```

```
*/
```

```
void add_to_pick_list (pick_list_type *list, char *item, int position) {
```

```
    strcpy (list->list[position-1], item);
```

```
    return ;
```

```
}
```


/*-----
MODULE: agrio.c

Preforms the sequential agreement data functions.

Written By : Greg McGregor

REVISION: What was revised?
- GMM 07-30-1991 NOTHING

-----*/

#include <stdio.h>
#include <ctype.h>
#include <bench.h>
#include <proc.io>
#include <sys\stat.h>
#include <windows.h>
#include <gkeys.h>
#include <misc.h>

#include <agreev3.h> /* struct formats */
#include <agreenum.h>
#include <control.h>
#include <phone.h>
#include <raperson.h>
#include <agrio.h>
#include <gbase.h>
#include <time.h>
#include <extnvar.h>

/*-----*/
/* Open the Primary Data File, its Indices */
/*-----*/

open_files()
{
 wintype win;

/*-----*/
/* make sure files exist */
/*-----*/

 iostat = stat("AGREENUM",&buf);
 if (iostat < 0) {
 win = note ("Error, open files(), ");
 gotoxy (1,3);
 cprintf ("NO AGREENUM file! IOSTAT = %d, Call Central Office",iostat);
 gotoxy (15,4);
 cprintf ("Press ESC to Exit!");
 getch();
 windowclose (win);
 exit(1);
 }

 iostat = stat("CONTROL",&buf);
 if (iostat < 0)
 {
 win = note ("Error, open_files(), ");
 gotoxy (1,3);
 cprintf ("NO Control file! IOSTAT = %d, Call Central Office",iostat);
 gotoxy (15,4);


```

        cprintf ("Press ESC to Exit!");
        getch();
        windowclose (win);
    }
    exit(1);
}
iostat = stat("PHONE",&buf);
if (iostat < 0)
{
    win = note ("Error, open_files(), ");
    gotoxy (1,3);
    cprintf ("NO Phone file! IOSTAT = %d, Call Central Office",iostat);
    gotoxy (15,4);
    cprintf ("Press ESC to Exit!");
    getch();
    windowclose (win);
    exit(1);
}
iostat = stat("RAPERSON",&buf);
if (iostat < 0)
{
    win = note ("Error, open_files(), ");
    gotoxy (1,3);
    cprintf ("NO R.A.P. file! IOSTAT = %d, Call Central Office",iostat);
    gotoxy (15,4);
    cprintf ("Press ESC to Exit!");
    getch();
    windowclose (win);
    exit(1);
}

/* ..... */
/*      open files      */
/* ..... */

ld_agreemnt = open_file9(FILE1, FSIZE1, UPDATE_MODE, keypos_agreemnt, FLDS_agreemnt,
agreemnt fld);

ld_control = open_file9(FILE2, FSIZE2, READ_MODE, keypos_control, FLDS_control, control_fld);

ld_phone = open_file9(FILE3, FSIZE3, UPDATE_MODE, keypos_phone, FLDS_phone, phone_fld);

ld_raperson = open_file9(FILE4, FSIZE4, READ_MODE, keypos_raperson, FLDS_raperson, raperson_fld);

ld_agreenum = open_file9(FILE10, FSIZE10, UPDATE_MODE, keypos_agreenum, FLDS_agreenum,
agreenum_fld);

iostat = reset_file9 (ld_control,&controlrec); /* load control file */
if (iostat < 0) {
    cerrtn ("ERROR: (open_files) Can't load control file!");
    exit (0);
}
iostat = reset_file9 (ld_agreenum,&agreenumrec);
if (iostat < 0) {
    cerrtn ("ERROR: (open_files) Can't load last agreement number!");
    exit (0);
}
new_agreeno = agreenumrec.last_agreement_number + 1;
sprintf(new_agreeno_a,"%08d", new_agreeno);

```



```

        strcpy(agreeno, new_agreeno_a);
/*
        strcat(agreeno, controlrec.origagency,4);
        if (controlrec.origagency[3] != '\0')
            strCHcat (agreeno, '\0');
*/
/*
        moveX (agrecmntrec.agreeno, agreeno, 13); */
        strcpy (agrecmntrec.agreeno, agrecno);

        moveX (agrecmntrec.origagency, controlrec.tau_id, 4);

```

```

    return(IOGOOD);
}

```

```

/* ..... */
/*      Close the Data Files      */
/* ..... */

```

```

close_a_file (fd)
int fd;
{
    close_file9(fd);
}

```

```

close_files()
{
    close_file9(fd_agrecmnt);
    close_file9(fd_control);
    close_file9(fd_phone);
    close_file9(fd_raperson);
    close_file9(fd_agrecnum);
}

```

```

/* ..... */
/*  add/update agreement record  */
/* ..... */

```

```

add_upd_agrecmnt(int p)
{
    wintype win;
    char ch;
    int iostat;
    static int agreement_added = 1;
    struct agreemnt_def temp_agrecmnt;

```

```

    null_field (&temp_agrecmnt, sizeof (temp_agrecmnt));
    moveX (temp_agrecmnt.agreeno, agreemntrec.agreeno, 12);
    selectinx9 (fd_agrecmnt, 1);
    iostat = exactkey9 (fd_agrecmnt, &temp_agrecmnt);
    if (iostat < 0) {
        agreement_added = 1; /* add it */
    } else agreement_added = 0;

```

```

    if (agrecmnt_added == 1) {
        derive_plugged_fields();
        iostat = addrec9(fd_agrecmnt, &agrecmntrec);
        if (iostat < 0) {
            win = note ("Error, add_upd_agrecmnt, ");

```



```

        gotoxy (1,3);
        cprintf ("fd_agreemnt 'add' IOSTAT = %d, Call Central Office",iostat);
        gotoxy (15,4);
        cprintf ("Press ESC to Exit!");
        getch();
        windowclose (win);
        close_files ();
        return;
    }
    agreement_added = 0;
    /* update control record with latest agreement number */

    agreenumrec.last agreement number = new agreeno;
    iostat = updrcc9(fd_agreenum, &agreenumrec);
    if (iostat < 0) {
        win = note ("ERROR: (add_upd_agreemnt)");
        gotoxy (1,3);
        cprintf ("fd_agreenum 'upd' IOSTAT = %d, Call Central Office",iostat);
        gotoxy (15,4);
        cprintf ("Press ESC to Exit!");
        getch();
        windowclose (win);
        close_files ();
        return ;
    }
} else {
    derive_plugged_fields();
    agreement_added = 1; /* reset for next rental */
    iostat = updrcc9(fd_agreemnt, &agreemntrec);
    if (iostat < 0) {
        win = note ("Error, add_upd_agreemnt, ");
        gotoxy (1,3);
        cprintf ("fd_agreemnt 'upd' IOSTAT = %d, Call Central Office",iostat);
        gotoxy (15,4);
        cprintf ("Press ESC to Exit!");
        getch();
        close_files ();
        return ;
    }
}

moveX (phonerec.curphoneno,agreemntrec.curphoneno,12);
if (p == 1) strcpy (phonerec.status,"1");
if (p == 2) strcpy (phonerec.status,"0"); /* closing agreement */
if (p == 3) strcpy (phonerec.status,"2"); /* Lost phone */
if (p == 4) strcpy (phonerec.status,"3"); /* Can't communicate to phone */
if (p == 5) strcpy (phonerec.status,"4"); /* Reserved */
iostat = updrcc9(fd_phone, &phonerec);
if (iostat < 0) {
    win = note ("Error, add_upd_agreemnt, ");
    gotoxy (1,3);
    cprintf ("fd_phone 'upd' IOSTAT = %d, Call Central Office",iostat);
    gotoxy (15,4);
    cprintf ("Press ESC to Exit!");
    getch();
    close_files ();
    return ;
}
}
}

```



```

/* ***** */
/* derive plugged fields */
/* ***** */

```

```

derive plugged fields()

```

```

{
    if (ESTIMATED_CALLS)
        update_tau_status (3,'6');
    lcopy(agreementrec.phoneid,phonerec.phoneid,12);
    /* moveX (agreementrec.curphoneno,phone_number,12); */
    agreementrec.phochgday = controlrec.phone_daily_chg;
    agreementrec.phochgmin = controlrec.charge_per_minute;

    switch (agreementrec.creditno[0]){
        case '3' : if (agreementrec.creditno[1] == '7')
                        strcpy (agreementrec.credittypc,"AE");
                    if (agreementrec.creditno[1] == '8')
                        strcpy (agreementrec.credittypc,"DC"); /* diners club
*/
                        break;
        case '4' : strcpy (agreementrec.credittypc,"VI");
                        break;
        case '5' : strcpy (agreementrec.credittypc,"MC");
                        break;
        case '6' : strcpy (agreementrec.credittypc,"DI"); /* discover */
                        break;
        case '9' : strcpy (agreementrec.credittypc,"CB");
                        break;
    }
    if (strcmp (agreementrec.approved,"CASH",4) == 0)
        strcpy (agreementrec.credittypc,"CA");
    if (strcmp (agreementrec.approved,"CHECK",5) == 0)
        strcpy (agreementrec.credittypc,"CK");
    if (strcmp (agreementrec.approved,"NONE",4) == 0)
        strcpy (agreementrec.credittypc,"NO");
}

```

```

swap_YM (agreementrec.expiredate);
}

```

```

swap_YM (a_date)

```

```

char a_date[];
{
    char _tmp[6];
    _tmp[0] = a_date[2];
    _tmp[1] = a_date[3];
    _tmp[2] = a_date[0];
    _tmp[3] = a_date[1];
    /* a_date = _tmp; */
}

```


-
- * MODULE: REALTIME.C
 - .
 - * TELEMAR CELLULAR CORP.
 - .
 - * Written By: Greg McGregor 1990
 - .
 - .
 - * PURPOSE: This module performs the real time billing event between
 - * the cellular phone and the hotel computer at Final Closing Time.

REVISED: What was revised?

GMM 7-30-1991 - The ability to estimate a call length is the
 battery on the phone was pulled so that no
 end of call data was stored.
 - Now gets cumulative air time reading every
 rental return.

GMM 9-13-1991 The real004 data was moved into the control file.
 Therefore, all references to real004 have changed to
 control.

```
#include <process.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <string.h>
#include <window.h>
#include <dos.h>
#include <bios.h>
#include <ctype.h>
#include <bench.h>
#include <proc.io>
#include <io.h>
#include <rtio.h>
#include <\sys\stat.h>

#include <gkeys.h>
#include <windows.h>
#include <agio.h>
#include <agreev3.h> /* all types, making them externs */
#include <control.h>
#include <phone.h>
#include <raperson.h>
#include <gbase.h>
#include <extnvar.h>
#include <extscns.h>
#include <rtb.h> /* realtime billing definitions */
#include <rtbfunc.h> /* common rtb functions */
#include <decphone.h> /* decode function to decode phone # */
#include <taustat.h>
#include <lostdam.h>
#include <cli_com.h>

#include <real001.h1> /* data records */
#include <real002.h1>
```



```

#include <real003.h1>
#include <real005.h1>

/*
 * Function Defs
 */
float rt_calc_billing(gbaserec rec);
bill_long_distance (record_type *call,float dist);
float calc_dist (record_type *call);
int set_inter (float d);
int set_intra (float d);
float time_to_seconds (char a_time[]);
char calc_rtb_LRC (char *t,int len);

#define BASE_POINTER 0x0F90 /* base pointer for novatel */
#define MAX_CALLS 100
#define MAX_STRING_LENGTH 11
#define CHART_MILE 6.25
long int DAY_RATE = 28800; /* day rate starts at 8am, in secs */
long int EVENING_RATE = 61200; /* evening starts at 5pm */
long int NIGHT_RATE = 82800; /* night starts at 11 pm */
int NO_SIGNAL = FALSE;
int MEMORY_FULL = FALSE;
int NO_BILL_UNDER;

char LRC_PHONE;

#define TIMED_OUT_X 1000 /* about 1 second time out */

/* chart_mile is what 1 unit converts to in miles */

#define COM1 0
#define COM2 1
#define COM3 2
#define COM4 3

#define CTI_BAUD 9600L
#define CTI_PORT COM1

/*-----
GLOBALS
-----*/
/* gbaserec call_rec; */ /* defined in extnvar.h */
/* contains agreeno and all calls asociated with
that agreeno. Times, dates, lengths etc.. */

record_type a_call; /* a call record that attaches to call_rec */

cti_obj eco; /* ending cti object */
char eco_buff[1024]; /* setup an eco buffer of 1K bytes */

/* all information is stored in the above two structures */
/* both are linked lists of records and are outputed that way to disk */
/* both defined in gbas.c */

char origin[5]; /* area code or origin of a call */
int METER_READING = 0; /* Cumulative air time meter in phone */
int NUM_ESTIMATE_CALLS = 0; /* Number of estimated calls - lengths of */

```



```

int no_bytes;

char far raw_data[8192]; /* ALLOWS FOR 8K OF DATA The phone can only */
/* store under 3K... 8K if for some reason a new phone can store more */
/*
char far raw_data[8192] = {0xF0,0x41,0x31,0x02,0x05,0x83,0x82,0x4A,0x10,0xE0,
                          0x41,0x31,0x09,0x07,0xF2,0x05,0x08,0x25,0x01,0x15,
                          0x13,0x68,0x94,0x56,0x70,0xE2,0x05,0x08,0x29,0x02,
                          0xF2,0x71,0x06,0x51,0x45,0xE2,0x71,0x06,0x59,0x01,0xF0};

*/

/*
* START PROGRAM CODE
*/

/**
gbaserec end_rtb ();

main ()
{
gbaserec rec1;
    rec1 = end_rtb();
}
**/

/*-----
parse_data ()
-----*/

int parse_data ()
{
char data;
int pos = 0;
int roamer_days = 0; /* # of 24hr periods phone was in roam */
int calls = 1;
int first_roam = TRUE;
record_type *call;
char roamer_date[8]; /* Used to calculate the roamer periods */

/* look for first call, in-use flag set */
while (move_hl (raw_data[pos]) != 0x0F) ++pos;
while (calls <= number_of_calls) {
    if ( (call = new_rec ()) == NULL ) {
        printf ("\n MALLOC: no memory!");
        printf ("\n      - Call: Telemac (800) 235-2356");
        exit (0);
    }
    if (!convert_call_info (call, &pos)) {
        rtb_error (-5);
        return FALSE;
    }
    moveX (call->tau_id,agrecmntrec.origagency,4);
    if (call->flag & 0x01) { /* roam light lit add in roaming charges */
        if (first_roam) {
            first_roam = FALSE;
            strncpy (roamer_date,call->date,6);
            ++roamer_days;
        } else
            if (strcmp (roamer_date,call->date,6) != 0) {
                strncpy (roamer_date,call->date,6);
                ++roamer_days;
            }
    }
}

```



```

        } /* else still in same day roaming */
    }
    assoc_rec (&call_rec, call);
    ++calls;
}
if (roamer_days != 0) {
    if ( (call = new_rec ()) == NULL ) {
        clrscr ();
        printf ("\n MALLOC: no memory!");
        printf ("\n      - Call: Telemac (800) 235-2356");
        exit (0);
    }
    strcpy (call->number, "ROAMING CHARGES");
    strcpy (call->start_time, "N/A");
    strcpy (call->end_time, "N/A");
    strncpy (call->date, agreemntrec.actrIndate, 6);
    call->length = roamer_days;
    call->actual_secs = 0.0;
    call->length_secs = 0.0;
    call->flag = 0x01; /* turn on bit 1 for roamer */
    call->next = NULL;
    add_in_roaming (call, roamer_days);
    assoc_rec (&call_rec, call);
    ++calls;
}
return TRUE;
}

```

```

/*-----
pre_parse_num_calls : parse data to determine number of calls
-----*/
int pre_parse_num_calls (int bytes) {
    int pos = 0;
    int calls = 0;
    char ch;
    int count = 0;
    while (pos < bytes) {
        ch = raw_data[pos];
        ch = move_hl(ch);
        if (ch == 0x0F) ++calls;
        ++pos;
        ++count;
        if (count >= 9) {
            clrscr ();
            cprintf ("-* Pre-Parsing Data - %d", pos);
            count = 0;
        }
    }
    clrscr ();
    cprintf ("-* Pre-Parsing Data - %d", pos);
    if (calls != 0)
        --calls; /* last data byte has F in upper nibble */
    return calls;
}

```

```

/*-----
add_in_roaming:
-----*/

```



```

int add_in_roaming (record_type *call, int days) {
int keynum;
    call->base_cost = days * controlrec.roam_chg_per_day;
    call->total_cost = call->base_cost;
    call->long_dist_cost = 0.0;
    return TRUE;
}

/*-----
convert_call_info: put call information into a record_type
    instead of breaking up into functions I prefer all of this garbage
    in one function. The data brought in is also variable length and
    type so it gets hairy below.
-----*/
int convert_call_info (record_type *call, int *pos)
{
char data_stream[50]; /* to store nibbles in low nibble of each byte */
char temp[20];
int i,j;
int roaming = FALSE;
time_t timer;
struct tm *tblock;
int EOC_FOUND = FALSE; /* End of call bool An 0xE found in high nibble */

    i = 0; /* init some fields */
    rtb_null_field (call->number, 40);

    if (move_hl (raw_data[*pos]) != 0x0F) return FALSE; /* 0x0F start of call */
    /* get call start data and put into stream in nibbles */

    do { /* get all of call info in a nibble stream */
        data_stream[i] = move_hl (raw_data[*pos]); /* move high nibble in */
        if (data_stream[i] == 0x0E)
            EOC_FOUND = TRUE; /* found start of end call data */
        ++i;
        data_stream[i] = raw_data[*pos] & 0x0F; /* Move lower nibble in */
        ++*pos;
        ++i;
    } while (move_hl (raw_data[*pos]) != 0x0F);
    /* just in case call doesn't end in 0x0E */
    data_stream[i] = 0x0F;

    /* first nibble is Hex F so skip start at 1 */
    temp[1] = to_digit (data_stream[1]); /* second digit of month */
    if (data_stream[2] & 0x04) {
        temp[0] = '1';
    } else temp[0] = '0'; /* moved in first digit of month based on flag */
    /* in 3rd nibble */

    /* check flags */
    call->flag = 0; /* null all flags in flag variable */
    if (data_stream[2] & 0x08) { /* bit 4 is set if roam light is on */
        call->flag = call->flag | 0x01; /* turn on bit 0 */
        roaming = TRUE;
    }

    /* move month into call rec */
    call->date[2] = temp[0]; /* date is stored as YYMMDD */
    call->date[3] = temp[1]; /* positions 012345 */
    /* now get year */
    timer = time (NULL);

```



```

tblock = (struct tm *)malloc (sizeof (struct tm) );
tblock = localtime (&timer);
strcpy (temp, itoa (tblock->tm_year, temp, 10));
free (tblock);
call->date[0] = temp[0];
call->date[1] = temp[1]; /* now year is moved in, year comes from */
/* the computers clock */

/* now get day of call */
temp[0] = data_stream[2] & 0x03; /* keep lower 2 bits */
call->date[4] = to_digit (temp[0]);
call->date[5] = to_digit (data_stream[3]);
call->date[6] = '\0'; /* null end of field */
/* date of call is now stored in call struct */

/* now get time of call */
if (data_stream[4] & BIT2){ /* am or pm */
    call->start_time[8] = 'P';
} else call->start_time[8] = 'A';
/* if over 10 and less=12 move a 1 into prev char for 10's place */
if ( (data_stream[5] >= 0x0A) && (data_stream[5] <= 0x0C) ){
    call->start_time[0] = '1';
    data_stream[5] -= 0x0A; /* subtract 10 */
    call->start_time[1] = to_digit (data_stream[5]);
} else {
    call->start_time[0] = '0';
    call->start_time[1] = to_digit (data_stream[5]);
}
call->start_time[2] = ':';
call->start_time[3] = to_digit (data_stream[6] & 0x07); /* keep 1st 3 bits */
call->start_time[4] = to_digit (data_stream[7]);
call->start_time[6] = to_digit (data_stream[8] & 0x07);
call->start_time[7] = to_digit (data_stream[9]);
call->start_time[5] = ':';
call->start_time[9] = '\0';
/* determine if call was made or call was answered */
strcpy (call->number, "INCOMING");
if (data_stream[10] != 0x0E) { /* call was made */
    rtb_null_field (call->number, 17);
    j = 0;
    do {
        if (data_stream[j+10] != 0) /* skip last 0 if exists */
            call->number[j] = to_digit (data_stream[j+10]);
        ++j;
    } while ( (data_stream[j+10] != 0x0E) &&
        (data_stream[j+10] != 0x0F) ); /* get number called */
    /* j+10 because next char may be 0E following the number */
    j = j + 10; /* jump to 0x0E */
} else j = 10;
if ( (data_stream[j] != 0x0E) &&
    (data_stream[j] != 0x0F) ) return FALSE; /* bad data transmit */

if (roaming)
    strcat (call->number, "");

if (IEOC_FOUND) { /* means 0x0F not a 0x0E above in bad data */
    call->flag = call->flag | 0x04;
    /* turn on estimate flag */
    ++NUM_ESTIMATE_CALLS; /* global */
    update_tau_status (3, '6');
}

```



```

    ESTIMATED_CALLS = TRUE;
    return TRUE;
}

/* get end time of call */
if (data_stream[j+4] & BIT2){ /* am or pm */
    call->end_time[8] = 'P';
} else call->end_time[8] = 'A';
/* if over 10 and less=12 move a 1 into prev char for 10's place */
if ( (data_stream[j+5] >= 0x0A) && (data_stream[j+5] <= 0x0C) ){
    call->end_time[0] = '1';
    data_stream[j+5] -= 0x0A; /* subtract 10 */
} else call->end_time[0] = '0';
call->end_time[1] = to_digit (data_stream[j+5]); /* set hour */
call->end_time[2] = ':';
call->end_time[3] = to_digit (data_stream[j+6] & 0x07); /* keep 1st 3 bits */
call->end_time[4] = to_digit (data_stream[j+7]);
call->end_time[6] = to_digit (data_stream[j+8] & 0x07);
call->end_time[7] = to_digit (data_stream[j+9]);
call->end_time[5] = ':';
call->end_time[9] = '\0';
/* get length of call */

calc_length_call (call);
}

/*-----
calc_length_call: calc the length of call
-----*/
calc_length_call (record_type *call)
{
    float start,end,total;
    char temp[10];
    int trunc_value;

    rtb_null_field (temp,10);
    temp[0] = call->start_time[0];
    temp[1] = call->start_time[1];
    temp[2] = '\0';

    if (strcmp (temp,"12",2) != 0) {
        start = (float)atoi (temp) * 3600; /* convert hours to seconds */
    } else start = 0; /* 12:00:00 is our starting position */

    temp[0] = call->start_time[3];
    temp[1] = call->start_time[4];
    start = start + (float)atoi (temp) * 60; /* convert mins to secs */
    temp[0] = call->start_time[6];
    temp[1] = call->start_time[7];
    start = start + (float)atoi (temp);

    rtb_null_field (temp,10);
    temp[0] = call->end_time[0];
    temp[1] = call->end_time[1];
    temp[2] = '\0';

    if (strcmp (temp,"12",2) != 0) {
        end = (float)atoi (temp) * 3600; /* convert hours to seconds */
    } else end = 0; /* 12:00:00 is our starting position */
}

```



```

temp[0] = call->end_time[3];
temp[1] = call->end_time[4];
end = end + (float)atoi(temp) * 60; /* convert mins to secs */
temp[0] = call->end_time[6];
temp[1] = call->end_time[7];
end = end + (float)atoi(temp);

if (call->start_time[8] == call->end_time[8]) {
    total = end - start;
} else {
    total = 12*3600 - start; /* time remaining in am or pm in secs */
    total = total + end;
}

call->length_secs = total; /* store in seconds format */
call->actual_secs = total;

/* bill INCOMING calls send to end */
if (strcmp(call->number, "INCOMING", 8) != 0)
    if (call->length_secs < NO_BILL_UNDER) {
        call->length_secs = 0;
    } else call->length_secs = call->length_secs;
/*
    } else call->length_secs = call->length_secs - NO_BILL_UNDER;
*/

    /* convert to minutes and round up */
    trunc_value = (int)(call->length_secs/60);
    if ((call->length_secs/60) > trunc_value) {
        total = trunc_value + 1; /* round up minutes */
    } else total = trunc_value;

    call->length = total; /* length stored now in minutes, adjusted */
}

/*-----
rt init databases:
-----*/
rt_init_databases()
{
    #include <real001.h2>
    #include <real002.h2>
    #include <real003.h2>
    #include <real005.h2>
}

/*-----
map_final    maps transmission for final rental
PARAMS: state
RETURNS: new state
PROCESS: state' = next(state) where next(state) = switch st. below
-----*/
int map_final(s)
int s; /* current state of transmission */
{
    switch (s) {
        case FINAL_STATE:
            return (GET_NUMBER);
        case GET_NUMBER:
            return (GET_POINTER);
    }
}

```



```

case GET_POINTER:
    return (GET_INFO);
case GET_INFO:
    return (NUMBER_CALLS);
case NUMBER_CALLS:
    return (READ_METER);
case READ_METER:
    return (LOCK_PHONE);
case LOCK_PHONE:
    return (POWER_DOWN);
case POWER_DOWN:
    return (END_STATE);
case END_STATE:
    return (END_STATE);
case ERROR_STATE: /* start over */
    return (FINAL_STATE);
}
return (ERROR_STATE); /* state doesn't exist */
}

/*-----
undo_return : turn on phone, unlock it and turn it off
-----*/
undo_return () {
int stat;
windef note_win = {10,7,70,9,White,Blue,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
                    White,Blue};
wintype note_wt;

note_wt = windowopen (&note_win);
settitle (note_wt,"Releasing Phone",CenterUpperTitle);
use (note_wt);
clrscr ();
cprintf ("          Setting Up Communication Port");
set_up_cti_object (&eco, CTI_BAUD, CTI_PORT, 1024, 2);
set_cti_buffer (&eco, &eco_buff); /* point sco buffer to sco buff */
stat = open_cti_port (&eco); /* start interrupts */

clrscr ();
cprintf ("          Turning On The Cellular Phone");
set_cti_command (&eco,TURN_ON);
stat = do_cti_command (&eco);
if (!stat) errrn ("ERROR (undo_return): Can't Turn On Phone ");
clrscr ();
cprintf ("          Unlocking The Cellular Phone");
set_cti_command (&eco,UNLOCK_PHONE);
stat = do_cti_command (&eco);
if (!stat) errrn ("ERROR (undo_return): Can't Unlock Phone ");
clrscr ();
cprintf ("          Turning The Cellular Phone OFF");
set_cti_command (&eco,POWER_DOWN);
stat = do_cti_command (&eco);
if (!stat) errrn ("ERROR (undo_return): Can't Turn Off Phone ");
windowclose (note_wt);

close_cti_port (&eco);
}

```



```

-----
end_rtb: Do final agreement processing billing
-----*/

end_rtb ()
{
int stat;
int fd;

ESTIMATED_CALLS = FALSE;
MEMORY_FULL = FALSE;
/* open data base file */
fd = g_open ("callrec.dat", O_RDWR, &call_rec);

set_origin (); /* set area code of phone, and connect time */

use (CTI_wt);
clrscr ();
cprintf ("      Opening Data Channel...");

set_up_cti_object (&eco, CTI_BAUD, CTI_PORT, 1024, 2);
set_cti_buffer (&eco, &eco_buff); /* point sco buffer to sco_buff */
stat = open_cti_port (&eco); /* start interrupts */

use (CTI_wt);
clrscr ();
cprintf ("      Starting CTI Retrieval Process...");
stat = final_transfer ();

close_cti_port (&eco);

if (stat <= 0) {
    call_rec.attached_records = stat;
    g_close (fd);
}
if (stat == -29) {
    call_rec.attached_records = -29;
    return -29;
}
if ( (stat != -25) && (stat != -26) && (stat != -21) ) {
    stat = damaged_phone_predicate2 ();
    if (stat == -1) {
        call_rec.attached_records = -26;
        return -26;
    }
    if (stat) {
        call_rec.attached_records = -25;
        return -25;
    }
}
return;
}

use (CTI_wt);
clrscr ();
cprintf ("      Parsing Data...");
if (!parse_data ()) {
    call_rec.attached_records = -4;
    g_close (fd);
    stat = damaged_phone_predicate2 ();
    if (stat == -1) {
        call_rec.attached_records = -26;
    }
}
}

```



```

        return -26;
    }
    if (stat) {
        call_rec.attached_records = -25;
        return -25;
    }
    return;
}

use (CTI_wt);
clrscr ();
cprintf ("    Calculating Bill... Call(#): ");
total rtb bill = rt calc billing (call_rec);

if (NUM_ESTIMATE_CALLS != 0)
    agrccmntrec.estimate_flag[0] = 'Y';
/* had to estimate some call lengths */
use (CTI_wt);
clrscr ();
cprintf ("    CTI Completed...");

g_close (fd);
}

/*-----
final_transfer
PARAMS: none
RETURNS 1 = success
        3 = data transmission error
        4 = phone not in box
PROCESS: gets call info, locks phone
-----*/

int final_transfer ()
{
    register int out,in,stat;
    int state = START STATE;
    int DONE = FALSE;
    int pos = 0;
    int trys;
    char abyte;
    char msg[80];
    abyte = 0x01;
    raw_data[0] = '\0';
    state = FINAL STATE;
    while (1) {
        /* determine state of transmission and do appropriate */
        switch (state) {
            case FINAL_STATE:
                use (CTI_wt);
                clrscr ();
                cprintf ("-* Checking For Cellular Phone");
                trys = 0;
                stat = FALSE;
                while ( (trys < 3) && (!stat) ) {
                    sct_cti_command (&eco,TURN_ON);
                    stat = do_cti_command (&eco);
                    if (!stat)
                        switch (trys) {
                            case 0:

```



```

stat = lost_phone_predicate ();
if (stat == -1) return -26;

break;

case 1:
stat = damaged_phone_predicate ();
if (stat == -1) return -26;
if (stat) return -25;
break;

case 2:
stat = damaged_phone_predicate ();
if (stat == -1) return -26;
if (stat) return -25;
break;
}

++trys;
}
state = map_final (state);
break;
case GET_NUMBER :
use (CTI wt);
clrscr ();
cprintf ("-* Retrieving Cellular Phone Number");
set_cti_command (&eco.GET_NUMBER); /* get data into sco_buff */
stat = do_cti_command (&eco);
decode_phone (agreemntrec.curphoneno,eco_buff);
if (!stat){
rtb_error (-9);
return -9;
};
decode_phone (agreemntrec.curphoneno,eco_buff);
/* see if phone is rented out or in #####*/
moveX (phonerec.curphoneno,agreemntrec.curphoneno,12);
stat = reset_file9 (fd_phone,&phonerec);
stat = selectinx9 (fd_phone,1);
moveX (phonerec.curphoneno,agreemntrec.curphoneno,12);
stat = exactkey9 (fd_phone,&phonerec);
if (stat < 0) {
rtb_error (-23);
call_rec.attached_records = -23;
return -23;
}
if (phonerec.status[0] != '1') {
phonerec.curphoneno[12] = '\0';
sprintf (msg,"This Phone, %s, has NOT been rented
out",phonerec.curphoneno);

errln (msg);
call_rec.attached_records = -21;
return -21;
}

state = map_final (state);
break;
case GET_POINTER :
use (CTI wt);
clrscr ();
cprintf ("-* Retrieving RTB memory pointer");
no_bytes = 0;
trys = 0;
stat = FALSE;
set_cti_command (&eco.GET_POINTER);
```



```

stat = do_cti_command (&eco);
    if (!stat){
        rtb_error (-20);
        return -20;
    }
    moveX (&no_bytes,eco_buff,2); /* move pointer to int */
if (no_bytes != BASE_POINTER) {
    no_bytes = no_bytes - BASE_POINTER + 1; /* +1 for LRC */
} else no_bytes = 0; /* make no_bytes = 0 */
    state = map_final (state);
    break;
case NUMBER_CALLS:
    trys = 0;
    clrscr ();
    cprintf ("-* Pre-Parsing Data -");
    stat = prc_parsc_num_calls (no_bytes);
if (stat < 0){
    rtb_error (-16);
    return -16;
}
    number of calls = stat;
    clrscr ();
    cprintf ("-* Calls : %d",stat);
    state = map_final (state);
    break;
case GET_INFO :
    use (CTL_wt);
    clrscr ();
    cprintf ("-* Retrieving Call DATA");
if (no_bytes != 0) {
    set_cti_command (&eco,GET_INFO);
    set_cti_rcc_count (&eco,no_bytes);
    set_cti_buffer (&eco,&raw_data);
    stat = do_cti_command (&eco);
    set_cti_buffer (&eco,&eco_buff);
} else stat = TRUE;
if (eco.rec_count got != 0) {
    LRC_PHONE = raw_data [eco.rec_count -1];
    if (LRC_PHONE != calc_rtb_LRC (raw_data,eco.rec_count -1)){
        rtb_error (-3);
        return -3;
    }
}
    if (!stat) {
        rtb_error (-3);
        return -3;
    }
if (no_bytes != 0)
    raw_data[eco.rec_count-1] = 0xF0; /* EOF char , overright LRC */
    state = map_final (state);
    break;
case READ_METER :
    use (CTL_wt);
    clrscr ();
    cprintf ("-* Retrieving Meter From Cellular Phone");
    stat = FALSE;
    set_cti_command (&eco,READ_METER);
    stat = do_cti_command (&eco);
    if (!stat){
        errrn ("Could NOT get meter reading from phone!");
    }

```



```

        rtb_error (-24);
        return -24;
    }
    METER_READING = 0;
    if ( (eco_buff[0] != 0) || (eco_buff[1] != 0) )
        ++METER_READING; /* secs if there is some round up to a minute */
    METER_READING += eco_buff[2]; /* add in minutes */
    METER_READING += (eco_buff[3] * 10); /* add in 10's of minutes */
    METER_READING += (eco_buff[4] * 60); /* add in hours in minutes */
    METER_READING += (eco_buff[5] * (60 * 10)); /* add in 10's of hours in
minutes */

    METER_READING += (eco_buff[6] * (60 * 100)); /* add in 100's of hours in mins
*/

    METER_READING += (eco_buff[7] * (60 * 1000)); /* add in 1000's of hours in
mins */

    state = map_final (state);
    break;
case LOCK_PHONE :
    use (CTI_wt);
    clrscr ();
    cprintf ("*- Locking Cellular Phone");
    set_cti_command (&eco, LOCK_PHONE);
    stat = do_cti_command (&eco);
    if (!stat) {
        rtb_error (-18);
        return -18;
    }
    state = map_final (state);
    break;
case POWER_DOWN :
    use (CTI_wt);
    clrscr ();
    cprintf ("*- Turning Cellular Phone OFF");
    set_cti_command (&eco, POWER_DOWN);
    stat = do_cti_command (&eco);
    if (!stat){
        rtb_error (-19);
        return -19;
    }
    state = map_final (state);
    break;
case END_STATE :
    return TRUE; /* finished OK */
    break;
case ERROR_STATE :
    wait_error ();
    rtb_error (-6);
    return -6;
    state = map_final (state);
    break;
}
}
return (TRUE);
}

```

```

/*-----
end_state
-----*/
end_state ()

```



```

{
}

/*-----
get number calls
-----*/
int get_number_calls ()
{
    int r_value = TRUE; /* return value */
    int stat;
    int count;
    int in;
    int t;
    char temp[10],ch;
    int start_address = 0x0F90;

    use (CTL_wl);
    wait_command ();
    clrscr ();
    cprintf (" - * Getting # Calls.");
    stat = bioscom (1, NUMBER CALLS, RTB_PORT);
    t = 0;
    count = 0;
    while ( (count < 2) && (t < TIMED_OUT_X) ){ /* 1 secnds */
        stat = bioscom (3,0,RTB_PORT);
        if (stat & DATA_READY) {
            in = bioscom (2,0,RTB_PORT);
            temp[count] = (char)in;
            ++count;
            t = 0;
        } else {
            wait_receive (); /* delay ms */
            ++t;
        }
    }
    temp[2] = '\0';

    if (t >= TIMED_OUT_X) {
        strcpy (temp, "\0\0");
        r_value = FALSE;
    }

    ch = temp[0]; /* swap bytes, to store properly in int type */
    temp[0] = temp[1];
    temp[1] = ch;

    moveX (&number_of_calls,temp,2);
    if (r_value) {
        clrscr ();
        cprintf (" - * Calls :%d",number_of_calls);
    } else {
        clrscr ();
        cprintf (" - * ERROR: Timed Out!");
    }
    return r_value;
}

```



```

/*-----
get_info
-----*/

int get_info (int bytes)
{
    int r_value = TRUE; /* return value */
    int stat;
    float tempf,tempf1;
    int tempint,tempint1;
    int count;
    int in;
    int t;
    char temp[10],ch;
    int start_address = 0x0F90;

    usc (CTI_wt);
    wait_command ();
    clrscr ();
    cprintf ("-* Receiving Call Data ->");
    stat = bioscom (1, GET_INFO, RTB_PORT);

    /*
    /* CTI firmware comp. 8-13-1991 */
    stat = bioscom (1, (char)(bytes & 0x00FF), RTB_PORT);
    stat = bioscom (1, (char)((bytes>>8) & 0x00FF), RTB_PORT);
    /* CTI firmware comp. 8-13-1991 */
    */

    t = 0;
    count = 0;
    while ( (count < bytes) && (t < TIMED_OUT_X) ){
        stat = bioscom (3,0,RTB_PORT);
        if (stat & DATA_READY) {
            in = bioscom (2,0,RTB_PORT);
            raw_data[count] = in;
            ++count;
            t = 0;
            gotoxy (28,1);
            tempf = (float)count;
            tempf1 = (float)bytes;
            if (tempint != tempint1)
                cprintf ("%d%% Complete",(int)((tempf/tempf1)*100.0));
            tempint = tempint1;
            tempint1 = (int)((tempf/tempf1)*100.0);
        } else {
            wait_receive ();
            ++t;
        }
    }

    if (bytes != 0) {
        LRC_PHONE = raw_data [count -1];
        if (LRC_PHONE != calc_rtb_LRC (raw_data,bytes -1))
            return FALSE;
    }
    raw_data[count-1] = 0xF0; /* EOF char , overright LRC*/
    if (t >= TIMED_OUT_X)
        return FALSE;
    return TRUE;
}

```



```

/*-----
calc_rtb_LRC
-----*/
char calc_rtb_LRC (char *t,int len)
{
    int i;
    char LRC;
        i = 0;
        LRC = t[i];
        i++;
        while (i<len){
            LRC = LRC^t[i];
            ++i;
        }
    return (LRC);
}

```

```

/*-----
get_bytes
-----*/
int get_bytes (int *bytes)
{
    int r_value = TRUE; /* return value */
    int stat;
    int count;
    int in;
    int t;
    char temp[10],ch;
    int start_address = 0x0F90;
    int end_address;

        use (CTL_wt);
        wait_command ();
        clrscr ();
        cprintf (" - * Getting # Bytes.");
        stat = bioscom (1, GET_POINTER, RTB_PORT);
        t = 0;
        count = 0;
        while ( (count < 2) && (t < TIMED_OUT_X) ){ /* 1 secnds */
            stat = bioscom (3,0,RTB_PORT);
            if (stat & DATA_READY) {
                in = bioscom (2,0,RTB_PORT);
                temp[count] = (char )in;
                ++count;
                t = 0;
            } else {
                wait_receive (); /* delay */
                ++t;
            }
        }

        temp[2] = '\0';

        if (t >= TIMED_OUT_X) {
            strcpy (temp,"\\0\\0");
            r_value = FALSE;
        }
}

```



```

moveX (bytes,temp,2);
if (r_value) {
    end_address = *bytes;
    if (*bytes == start_address) {
        *bytes = 0;
    } else *bytes = *bytes - start_address + 1; /* +1 FOR THE LRC */
    clrscr ();
    printf ("-* Bytes :%d",*bytes);
} else {
    clrscr ();
    printf ("-* ERROR :Timed Out");
}
if (end_address >= 0x1A00) /* if memory full, run off of meter */
    MEMORY_FULL = TRUE;
return r_value;
}

```

```

/*-----
open rt files
-----*/

```

```

int open_rt_files()

```

```

{

```

```

    wintype win;

```

```

    int iostat;

```

```

    iostat = stat("REAL001",&buf);

```

```

    if (iostat < 0)

```

```

        return FALSE;

```

```

    iostat = stat ("REAL002",&buf);

```

```

    if (iostat < 0)

```

```

        return FALSE;

```

```

    iostat = stat ("REAL003",&buf);

```

```

    if (iostat < 0)

```

```

        return FALSE;

```

```

    iostat = stat ("REAL005",&buf);

```

```

    if (iostat < 0)

```

```

        return FALSE;

```

```

        // make sure controlrec is in memory

```

```

    iostat = reset_file9 (fd_control,&controlrec);

```

```

    if (iostat < 0)

```

```

        return FALSE;

```

```

if ((fd_real001= open_file9 (FILE5, FSIZE5, READ_MODE,keypos_real001,
    FLDS_real001, real001_fid)) < IOGOOD)

```

```

    return FALSE;

```

```

if ((fd_real005= open_file9 (FILE9, FSIZE9, READ_MODE,keypos_real005,
    FLDS_real005, real005_fid)) < IOGOOD)

```

```

    return FALSE;

```

```

if ((fd_real002= open_file9 (FILE6, FSIZE6, READ_MODE,keypos_real002,
    FLDS_real002, real002_fid)) < IOGOOD)

```

```

    return FALSE;

```

```

if ((fd_real003= open_file9 (FILE7, FSIZE7, READ_MODE,keypos_real003,
    FLDS_real003, real003_fid)) < IOGOOD)

```



```

        return FALSE;

    return TRUE; /* all databascs ok and exist */
}

/*-----
close_rt_files ()
-----*/
close_rt_files ()
{
    close_file9 (fd_real001);
    close_file9 (fd_real002);
    close_file9 (fd_real003);
    close_file9 (fd_real005);
}

/*-----
rt_calc_total; Calc the total charge for calls;
-----*/
float rt_calc_total(gbaserec rec)
{
    int i;
    float total;
    record_type *call;

    total = 0;
    base_cost = 0;
    long_dist = 0;
    number of calls = rec.attached_records;
    i = 1;
    if (rec.attached_records == 0)
        return 0;
    while (i <= rec.attached_records) {
        call = g_get_call (rec,i);
        base_cost += call->base_cost;
        long_dist += call->long_dist_cost;
        total = total + call->total_cost;
        ++i;
    }
    return total;
}

/*-----
total_real_minutes : add up total usage time base on clock chip
-----*/
int total_real_minutes (gbaserec rec) {
    int i;
    int total;
    record_type *call;

    total = 0;
    i = 1;
    if (rec.attached_records == 0)
        return 0;
    while (i <= rec.attached_records) {
        call = g_get_call (rec,i);
        total += (int)call->length;
        ++i;
    }
    return total;
}

```


}

```

=====
rt calc_billing() : Calculate charges on every call
=====
float rt_calc_billing(gbaserec rec)
{
    int i, call_type = -1; /* 1 local, 2 long dist, 3 International */
    int real_minutes = 0; /* total minutes used */
    int ave_minutes = 1; /* default estimate to 1 minute */
    record_type *call;

    i = 1;
    set_origin();
    /* 'Additional' means calls were based off of meter */
    /* because too many were made. Don't need to calc them */
    if (NUM_ESTIMATE_CALLS != 0) {
        real_minutes = total_real_minutes(rec);
        if (METER_READING > real_minutes) {
            ave_minutes = (METER_READING - real_minutes) / NUM_ESTIMATE_CALLS;
        }
    }
    while ((i <= rec.attached_records) &&
           (strcmp(call->number, "Additional", 10) != 0)) {

        NO_SIGNAL = FALSE;
        gotoxy(39, 1);
        cprintf("%d", i);
        call = g_get_call(rec, i);
        call->long_dist_cost = 0; /* init charges */
        call->base_cost = 0;
        call->total_cost = 0;

        call_type = 1; /* base case is call_type local */

        if ((call->number[0] == '1') && (strlen(call->number) > 7)) {
            if ((call->number[2] == '0') || (call->number[2] == '1')) {
                call_type = 2;
                shift_left(call->number, 20); /* delete the 1 */
                if (is_in_real001(call))
                    call_type = 3; /* international default */
                if (is_in_control(call))
                    call_type = 1; /* dialed area code for local call */
            }
        }

        if (((call->number[1] == '0') || (call->number[1] == '1'))
            && (strlen(call->number) > 7)) {
            call_type = 2;
            if (is_in_real001(call))
                call_type = 3;
            if (is_in_control(call))
                call_type = 1; /* dialed area code for local call */
        }

        if (call->number[0] == '0')
            call_type = 1; /* credit card call */

        if (strcmp(call->number, "01", 2) == 0) {

```



```

        call_type = 1; /* operator assistance */
    }

    if (strcmp (call->number,"011",3) == 0) {
        call_type = 3; /* direct out of country call */
    }

    if (strcmp (call->number,"800",3) == 0)
        call_type = 1;

    if (strcmp (call->number,"INCOMING",8) == 0)
        call_type = 1;

    if (call->flag & 0x02) /* no signal */
        NO_SIGNAL = TRUE;

    if (call->flag & 0x04) /* estimate length of call */
        call->length = ave_minutes;
        call->length_secs = call->length * 60;
    }

    /* SEE functions bill_local, bill_longdist, bill_inter etc..
       if (call->flag & 0x01) /* roam light lit, bill longdistance */
       call_type = 2;
    */

    /* if length of call = 0 do not bill, adjusted to 0 in */
    /* in parsing above in function calc call length ? */

    if ( (!NO_SIGNAL) && (call->length_secs != 0) ){
        if (call_type == 1)
            bill_local (call);
        if (call_type == 2) {
            if (call->flag & 0x04) {
                bill_long_distance (call,5700);
            } else
                bill_long_distance (call,calc_dist (call));
        }
        if (call_type == 3 )
            bill_out_country (call);
    } else
        if (strcmp (call->number,"ROAM",4) == 0) {
            add_in_roaming (call,call->length);
            /* call->length refers to # of days in roam, only here */
        } else {
            call->total_cost = 0; /* no charge for call */
        }
    }
    ++i;
}
return rt_calc_total (rec);
}

```

```

/*-----
set_origin: set the origin of the call made
-----*/
set_origin ()
{
    int found;

```



```

found = FALSE;
strcpy (origin,controlrec.area_code_of_tau,3);
NO_BILL_UNDER = controlrec.del_billing_secs; /* set connect time */
}

```

```

/*=====
is_in_control : is call in data base real004, a local call

```

```

9-13-1991      This has modified to really get data from the control file

```

```

/*=====

```

```

int is_in_control (record_type *call)

```

```

{
int found;
found = FALSE;
/* compare area codes */
if (strcmp (controlrec.area_code_of_tau,call->number,3) == 0) {
found = TRUE;}
if (strcmp (controlrec.local_area_code1,call->number,3) == 0) {
found = TRUE;}
if (strcmp (controlrec.local_area_code2,call->number,3) == 0) {
found = TRUE;}
if (strcmp (controlrec.local_area_code3,call->number,3) == 0) {
found = TRUE;}
if (strcmp (controlrec.local_area_code4,call->number,3) == 0) {
found = TRUE;}

```

```

return ( found );
}

```

```

/*-----
is_in_real005: is call intraState -- Instate call
-----*/

```

```

int is_in_real005 (record_type *call)

```

```

{
int stat,keynum;
struct real005_def temp005rec;
int found;

found = FALSE;
selectinx9 (fd_real005,keynum);
stat = reset_file9 (fd_real005,&real005rec);

```

```

strcpy (real005rec.mainarea,origin,3);

```

```

stat = exactkey9 (fd_real005,&real005rec);

```

```

if (strcmp (real005rec.mainarea,call->number,3) == 0)
found = TRUE;
if (strcmp (real005rec.area1,call->number,3) == 0)
found = TRUE;
if (strcmp (real005rec.area2,call->number,3) == 0)
found = TRUE;
if (strcmp (real005rec.area3,call->number,3) == 0)
found = TRUE;
if (strcmp (real005rec.area4,call->number,3) == 0)
found = TRUE;
if (strcmp (real005rec.area5,call->number,3) == 0)
found = TRUE;

```



```

if (strcmp (real005rec.area6,call->number,3) == 0)
    found = TRUE;
if (strcmp (real005rec.area7,call->number,3) == 0)
    found = TRUE;
if (strcmp (real005rec.area8,call->number,3) == 0)
    found = TRUE;
if (strcmp (real005rec.area9,call->number,3) == 0)
    found = TRUE;
if (strcmp (real005rec.area10,call->number,3) == 0)
    found = TRUE;
if (strcmp (real005rec.area11,call->number,3) == 0)
    found = TRUE;
if (strcmp (real005rec.area12,call->number,3) == 0)
    found = TRUE;
return found;
}

```

```

/*=====
is_in_real001 : is area code in list of documented long dist area cds.
=====*/

```

```

int is_in_real001 (record_type *call)
{

```

```

    int stat,keynum;
    struct real001_dcf temp001rec;
    int found;
    char temp[4];

```

```

    found = FALSE;

```

```

    selectinx9 (fd_real001,1);
    stat = reset_file9 (fd_real001,&real001rec);

```

```

    temp001rec = real001rec;
        /* compare area codes */
    temp [0] = call->number[0]; /* the 1 is nuked in calling procedure */
    temp [1] = call->number[1];
    temp [2] = call->number[2];
    temp [3] = '\0';
    moveX (temp001rec,temp,3);
    stat = exactkey9 (fd_real001,&temp001rec);
    if (stat < IOGOOD) {
        found = FALSE;
    } else found = TRUE;
    return found;
}

```

```

/*=====
under_time : is call under billable time, NO_BILL_UNDER
=====*/

```

```

int under_time (record_type *call)
{
    if (call->length_secs < NO_BILL_UNDER)
        return (TRUE);
    return (FALSE);
}

```

```

/*=====
bill_local : bill for a local call
=====*/
bill_local (record_type *call)

```



```
{
int stat,keynum;
int trunc_valuc;
```

```
/* do calculations here */
```

```
call->total_cost = call->length * controlrec.charge_per_minute;
if (call->flag & 0x01) /* roam light lit add in roaming charges */
    call->total_cost += call->length * controlrec.roam_chg_per_min;
call->base_cost = call->total_cost;
}
```

```
/*=====
bill_long_distance : bill for a long-distance call
=====*/
```

```
bill_long_distance (record_type *call,float dist)
```

```
{
int stat,stat2,dist_type;
struct real002_def temp002rec;
struct real003_def temp003rec;
struct real005_def temp005rec;
```

```
selectinx9 (fd_real002,1);
stat = reset_file9 (fd_real002,&temp002rec);
selectinx9 (fd_real003,1);
stat = reset_file9 (fd_real003,&temp003rec);
selectinx9 (fd_real005,1);
stat2 = reset_file9 (fd_real005,&temp005rec);
```

```
/* do calculations here */
```

```
call->base_cost = call->length * controlrec.charge_per_minute;
```

```
if (call->flag & 0x01) /* roam light lit add in roaming charges */
    call->total_cost += call->length * controlrec.roam_chg_per_min;
```

```
call->long_dist_cost = 0; /* init long dist */
```

```
if (is_in_real005 (call)) { /* IntraState call, INSTATE */
```

```
    /* use real002 for intrastate charges */
```

```
    dist_type = set_intra (dist);
```

```
    switch (dist_type) {
```

```
        case 1 : /* 0 to 20 */
```

```
            /* night time 11pm to 8 am */
```

```
            if ( ((time_to_seconds (call->start_time)) <= DAY_RATE) ||
                ((time_to_seconds (call->start_time)) >= NIGHT_RATE) ){
                call->long_dist_cost = temp002rec.x0to20ni +
                    (temp002rec.x0to20na * (call->length - 1));
```

```
            } else /* day */
```

```
            if ( (time_to_seconds (call->start_time)) < EVENING_RATE) {
                call->long_dist_cost = temp002rec.x0to20di +
                    (temp002rec.x0to20da * (call->length - 1));
```

```
            } else
```

```
            if ( (time_to_seconds (call->start_time)) < NIGHT_RATE) {
                call->long_dist_cost = temp002rec.x0to20ei +
                    (temp002rec.x0to20ea * (call->length - 1));
```

```
            }
```

```
        break;
```



```

case 2: /* 21 to 40 miles */
    if ( ((time_to_seconds (call->start_time)) < DAY_RATE) ||
          ((time_to_seconds (call->start_time)) >= NIGHT_RATE) ){
        call->long_dist_cost = temp002rec.x21to40ni +
            (temp002rec.x21to40na * (call->length - 1));
    } else /* day */
        if ( (time_to_seconds (call->start_time)) < EVENING_RATE) {
            call->long_dist_cost = temp002rec.x21to40di +
                (temp002rec.x21to40da * (call->length - 1));
        } else
            if ( (time_to_seconds (call->start_time)) < NIGHT_RATE) {
                call->long_dist_cost = temp002rec.x21to40ei +
                    (temp002rec.x21to40ea * (call->length - 1));
            }
        break;
case 3: /* 41 to 70 miles */
    if ( ((time_to_seconds (call->start_time)) < DAY_RATE) ||
          ((time_to_seconds (call->start_time)) >= NIGHT_RATE) ){
        call->long_dist_cost = temp002rec.x41to70ni +
            (temp002rec.x41to70na * (call->length - 1));
    } else /* day */
        if ( (time_to_seconds (call->start_time)) < EVENING_RATE) {
            call->long_dist_cost = temp002rec.x41to70di +
                (temp002rec.x41to70da * (call->length - 1));
        } else
            if ( (time_to_seconds (call->start_time)) < NIGHT_RATE) {
                call->long_dist_cost = temp002rec.x41to70ei +
                    (temp002rec.x41to70ea * (call->length - 1));
            }
        break;
case 4: /* 71 to 100 miles */
    if ( ((time_to_seconds (call->start_time)) < DAY_RATE) ||
          ((time_to_seconds (call->start_time)) >= NIGHT_RATE) ){
        call->long_dist_cost = temp002rec.x71to100ni +
            (temp002rec.x71to100na * (call->length - 1));
    } else /* day */
        if ( (time_to_seconds (call->start_time)) < EVENING_RATE) {
            call->long_dist_cost = temp002rec.x71to100di +
                (temp002rec.x71to100da * (call->length - 1));
        } else
            if ( (time_to_seconds (call->start_time)) < NIGHT_RATE) {
                call->long_dist_cost = temp002rec.x71to100ei +
                    (temp002rec.x71to100ea * (call->length - 1));
            }
        break;
case 5: /* 101 to 150 miles */
    if ( ((time_to_seconds (call->start_time)) < DAY_RATE) ||
          ((time_to_seconds (call->start_time)) >= NIGHT_RATE) ){
        call->long_dist_cost = temp002rec.x101t150ni +
            (temp002rec.x101t150na * (call->length - 1));
    } else /* day */
        if ( (time_to_seconds (call->start_time)) < EVENING_RATE) {
            call->long_dist_cost = temp002rec.x101t150di +
                (temp002rec.x101t150da * (call->length - 1));
        } else
            if ( (time_to_seconds (call->start_time)) < NIGHT_RATE) {
                call->long_dist_cost = temp002rec.x101t150ei +
                    (temp002rec.x101t150ea * (call->length - 1));
            }
        break;

```



```

case 6: /* 151 to 330 miles */
    if ( ((time_to_seconds (call->start_time)) < DAY_RATE) ||
          ((time_to_seconds (call->start_time)) >= NIGHT_RATE) ){
        call->long_dist_cost = temp002rec.x151t330ni +
                               (temp002rec.x151t330na * (call->length - 1));
    } else /* day */
    if ( (time_to_seconds (call->start_time)) < EVENING_RATE) {
        call->long_dist_cost = temp002rec.x151t330di +
                               (temp002rec.x151t330da * (call->length - 1));
    } else
    if ( (time_to_seconds (call->start_time)) < NIGHT_RATE) {
        call->long_dist_cost = temp002rec.x151t330ei +
                               (temp002rec.x151t330ca * (call->length - 1));
    }
    break;
case 7: /* over 330 miles */
    if ( ((time_to_seconds (call->start_time)) < DAY_RATE) ||
          ((time_to_seconds (call->start_time)) >= NIGHT_RATE) ){
        call->long_dist_cost = temp002rec.xover330ni +
                               (temp002rec.xover330na * (call->length - 1));
    } else /* day */
    if ( (time_to_seconds (call->start_time)) < EVENING_RATE) {
        call->long_dist_cost = temp002rec.xover330di +
                               (temp002rec.xover330da * (call->length - 1));
    } else
    if ( (time_to_seconds (call->start_time)) < NIGHT_RATE) {
        call->long_dist_cost = temp002rec.xover330ei +
                               (temp002rec.xover330ca * (call->length - 1));
    }
    break;
} /* end switch */
} else { /* if intra state calcs */
    /* interstate mainland calls */
    dist_type = set_inter (dist);
    switch (dist_type) {
        case 1: /* 0 to 10 miles */
            if ( ((time to seconds (call->start time)) < DAY RATE) ||
                  ((time to seconds (call->start time)) >= NIGHT_RATE) ){
                call->long_dist_cost = temp003rec.x0to10ni +
                                         (temp003rec.x0to10na * (call->length - 1));
            } else /* day */
            if ( (time_to_seconds (call->start_time)) < EVENING_RATE) {
                call->long_dist_cost = temp003rec.x0to10di +
                                         (temp003rec.x0to10da * (call->length - 1));
            } else
            if ( (time to seconds (call->start time)) < NIGHT_RATE) {
                call->long_dist_cost = temp003rec.x0to10ei +
                                         (temp003rec.x0to10ca * (call->length - 1));
            }
            break;
        case 2: /* 11 to 22 miles */
            if ( ((time_to_seconds (call->start_time)) < DAY_RATE) ||
                  ((time to seconds (call->start time)) >= NIGHT RATE) ){
                call->long_dist_cost = temp003rec.x11to22ni +
                                         (temp003rec.x11to22na * (call->length - 1));
            } else /* day */
            if ( (time_to_seconds (call->start_time)) < EVENING_RATE) {
                call->long_dist_cost = temp003rec.x11to22di +
                                         (temp003rec.x11to22da * (call->length - 1));
            } else
    
```



```

if ( (time_to_seconds (call->start_time)) < NIGHT_RATE) {
    call->long_dist_cost = temp003rec.x11to22ei +
        (temp003rec.x11to22ca * (call->length - 1));
}
break;
case 3: /* 23 to 55 miles */
if ( ((time_to_seconds (call->start_time)) < DAY_RATE) ||
    ((time_to_seconds (call->start_time)) >= NIGHT_RATE) ){
    call->long_dist_cost = temp003rec.x23to55ni +
        (temp003rec.x23to55na * (call->length - 1));
} else /* day */
if ( (time_to_seconds (call->start_time)) < EVENING_RATE) {
    call->long_dist_cost = temp003rec.x23to55di +
        (temp003rec.x23to55da * (call->length - 1));
} else
if ( (time_to_seconds (call->start_time)) < NIGHT_RATE) {
    call->long_dist_cost = temp003rec.x23to55ei +
        (temp003rec.x23to55ea * (call->length - 1));
}
break;
case 4: /* 56 to 124 miles */
if ( ((time_to_seconds (call->start_time)) < DAY_RATE) ||
    ((time_to_seconds (call->start_time)) >= NIGHT_RATE) ){
    call->long_dist_cost = temp003rec.x56to124ni +
        (temp003rec.x56to124na * (call->length - 1));
} else /* day */
if ( (time_to_seconds (call->start_time)) < EVENING_RATE) {
    call->long_dist_cost = temp003rec.x56to124di +
        (temp003rec.x56to124da * (call->length - 1));
} else
if ( (time_to_seconds (call->start_time)) < NIGHT_RATE) {
    call->long_dist_cost = temp003rec.x56to124ei +
        (temp003rec.x56to124ea * (call->length - 1));
}
break;
case 5: /* 125 to 292 miles */
if ( ((time_to_seconds (call->start_time)) < DAY_RATE) ||
    ((time_to_seconds (call->start_time)) >= NIGHT_RATE) ){
    call->long_dist_cost = temp003rec.x125t292ni +
        (temp003rec.x125t292na * (call->length - 1));
} else /* day */
if ( (time_to_seconds (call->start_time)) < EVENING_RATE) {
    call->long_dist_cost = temp003rec.x125t292di +
        (temp003rec.x125t292da * (call->length - 1));
} else
if ( (time_to_seconds (call->start_time)) < NIGHT_RATE) {
    call->long_dist_cost = temp003rec.x125t292ei +
        (temp003rec.x125t292ca * (call->length - 1));
}
break;
case 6: /* 293 to 430 miles */
if ( ((time_to_seconds (call->start_time)) < DAY_RATE) ||
    ((time_to_seconds (call->start_time)) >= NIGHT_RATE) ){
    call->long_dist_cost = temp003rec.x293t430ni +
        (temp003rec.x293t430na * (call->length - 1));
} else /* day */
if ( (time_to_seconds (call->start_time)) < EVENING_RATE) {
    call->long_dist_cost = temp003rec.x293t430di +
        (temp003rec.x293t430da * (call->length - 1));
} else

```



```

if ( (time_to_seconds (call->start_time)) < NIGHT_RATE) {
    call->long_dist_cost = temp003rec.x293t430ei +
        (temp003rec.x293t430ea * (call->length - 1));
}
break;
case 7: /* 431 miles to 925 */
    if ( ((time_to_seconds (call->start_time)) < DAY_RATE) ||
        ((time_to_seconds (call->start_time)) >= NIGHT_RATE) ){
        call->long_dist_cost =
temp003rec.x431t925ni +
            (temp003rec.x431t925na * (call->length - 1));
    } else /* day */
    if ( (time_to_seconds (call->start_time)) < EVENING_RATE) {
        call->long_dist_cost = temp003rec.x431t925di +
            (temp003rec.x431t925da * (call->length - 1));
    } else
    if ( (time_to_seconds (call->start_time)) < NIGHT_RATE) {
        call->long_dist_cost = temp003rec.x431t925ei +
            (temp003rec.x431t925ea * (call->length - 1));
    }
    break;
case 8: /* 926 to 1910 */
    if ( ((time_to_seconds (call->start_time)) < DAY_RATE) ||
        ((time_to_seconds (call->start_time)) >= NIGHT_RATE) ){
        call->long_dist_cost = temp003rec.x925t191ni +
            (temp003rec.x925t191na * (call->length - 1));
    } else /* day */
    if ( (time_to_seconds (call->start_time)) < EVENING_RATE) {
        call->long_dist_cost = temp003rec.x925t191di +
            (temp003rec.x925t191da * (call->length - 1));
    } else
    if ( (time_to_seconds (call->start_time)) < NIGHT_RATE) {
        call->long_dist_cost = temp003rec.x925t191ei +
            (temp003rec.x925t191ea * (call->length - 1));
    }
    break;
case 9: /* 1910 to 3000 miles */
    if ( ((time_to_seconds (call->start_time)) < DAY_RATE) ||
        ((time_to_seconds (call->start_time)) >= NIGHT_RATE) ){
        call->long_dist_cost = temp003rec.x191t300ni +
            (temp003rec.x191t300na * (call->length - 1));
    } else /* day */
    if ( (time_to_seconds (call->start_time)) < EVENING_RATE) {
        call->long_dist_cost = temp003rec.x191t300di +
            (temp003rec.x191t300da * (call->length - 1));
    } else
    if ( (time_to_seconds (call->start_time)) < NIGHT_RATE) {
        call->long_dist_cost = temp003rec.x191t300ei +
            (temp003rec.x191t300ea * (call->length - 1));
    }
    break;
case 10: /* 3001 to 4250 miles */
    if ( ((time_to_seconds (call->start_time)) < DAY_RATE) ||
        ((time_to_seconds (call->start_time)) >= NIGHT_RATE) ){
        call->long_dist_cost = temp003rec.x301t425ni +
            (temp003rec.x301t425na * (call->length - 1));
    } else /* day */
    if ( (time_to_seconds (call->start_time)) < EVENING_RATE) {
        call->long_dist_cost = temp003rec.x301t425di +
            (temp003rec.x301t425da * (call->length - 1));
    }

```



```

    } else
    if ( (time_to_seconds (call->start_time)) < NIGHT_RATE) {
        call->long_dist_cost = temp003rec.x301t425ci +
            (temp003rec.x301t425ea * (call->length - 1));
    }
    break;
case 11 : /* 4251 to 5750 */
    if ( ((time_to_seconds (call->start_time)) < DAY_RATE) ||
        ((time_to_seconds (call->start_time)) >= NIGHT_RATE) ){
        call->long_dist_cost = temp003rec.x425t575ni +
            (temp003rec.x425t575na * (call->length - 1));
    } else /* day */
    if ( (time_to_seconds (call->start_time)) < EVENING_RATE) {
        call->long_dist_cost = temp003rec.x425t575di +
            (temp003rec.x425t575da * (call->length - 1));
    } else
    if ( (time_to_seconds (call->start_time)) < NIGHT_RATE) {
        call->long_dist_cost = temp003rec.x425t575ei +
            (temp003rec.x425t575ca * (call->length - 1));
    }
    break;
} /* end switch */
} /* end else and interstate mainland longdistance charges */

call->long_dist_cost = (call->long_dist_cost * (controlrec.long_dist_markup + 1));
/* make it 1.xx percent >>> HERE ^^^ */
round_f (&call->long_dist_cost); /* round cost */
call->total_cost = call->base_cost + call->long_dist_cost;
/* and wall-a a bill is created yeah! */
}

/*-----
set_intra : set intrastate call code
-----*/
int set_intra (float d)
{
    if (d <= 20)
        return 1;
    if (d <= 40)
        return 2;
    if (d <= 70)
        return 3;
    if (d <= 100)
        return 4;
    if (d <= 150)
        return 5;
    if (d <= 330)
        return 6;
    return 7; /* over 330 miles */
}

/*-----
set_inter : set interstate call code
-----*/
int set_inter (float d)
{
    if (d <= 10)
        return 1;
    if (d <= 22)
        return 2;

```



```

    if (d <= 55)
        return 3;
    if (d <= 124)
        return 4;
    if (d <= 292)
        return 5;
    if (d <= 430)
        return 6;
    if (d <= 925)
        return 7;
    if (d <= 1910)
        return 8;
    if (d <= 3000)
        return 9;
    if (d <= 4250)
        return 10;
    if (d <= 5750);
        return 11;
    return 11; /* should never get here */
}

/*=====
bill_out_country
=====*/
bill_out_country (record_type *call)
{
    int stat;
    call->total_cost = call->length * controlrec.int_minute_rate +
        call->length * controlrec.roam_chg_per_min;
    /* add in roamer charges even on international calls */
    call->base_cost = call->total_cost;
}

/*=====
calc_dist : calculate distance of call ;
=====*/
float calc_dist (record_type *call)
{
    int stat, keynum;
    struct real001_def temp001rec, from, to;
    int found, found2;
    char temp[4];
    float dist;

    found = FALSE;
    found2 = FALSE;

    selectinx9 (fd_real001, 1);
    stat = reset_file9 (fd_real001, &real001rec);

    temp001rec = real001rec;
    /* compare area codes */
    moveX (temp001rec.areacode, origin, 3);
    stat = exactkey9 (fd_real001, &temp001rec);
    if (stat >= IOGOOD) found = TRUE;
    from = temp001rec;

    temp[0] = call->number[0]; /* 1 stripped earlier in front of call */
    temp[1] = call->number[1];

```


50

```

temp [2] = call->number[2];
temp [3] = '\0';
moveX (real001rec.arcacode,temp,3);
stat = exactkey9 (fd_real001,&real001rec);
if (stat >= IOGOOD) {
    if (found) found2 = TRUE;
}
to = real001rec;

/* means it found both area codes */
if (found2 == TRUE) {
    dist = CHART_MILE * sqrt (
        ((from.coordx - to.coordx) *
         (from.coordx - to.coordx)) +
        ((from.coordy - to.coordy) *
         (from.coordy - to.coordy)) );
    return (dist);
}
/* dist = CHART_MILE * SQAURE-ROOT ( (x1-x)^2 + (y1-y)^2); */
return (5700); /* problem return */
}

```


CREDIT.C

```

/*-----
MODULE :Credit.c                                VERSION 2.01

Credit Authorization Module : CREDIT    11/90

Written By : Greg McGregor 1990

REVISION:                                What was revised?
- GMM 7-30-1991                          Nothing
- GMM 8-10-1991+                         Reserve money and push it through
-----*/

```

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <time.h>
#include <window.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include "asiports.h"
#include "ibmkeys.h"
#include "gf.h"

```

```

#include <windows.h>
#include <gkeys.h>
#include <gbase.h>
#include <extnvar.h>
#include <gstring.h>
#include <taustat.h>

```

```
wintype c_win,auth_wt;
```

```

#define FULL 1 /* duplex definition */
#define HALF 0
#define MODE ASINOUT|BINARY|NORMALRX /* BINARY OR ASCII, Binary has
8
sig bits*/
#define RXLEN 1000 /* Receive buffer size */
#define TXLEN 1000 /* Transfer buffer size */
#define ECHO 0 /* 0 off, 1 on */
#define SPEAKER OFF
#define RTS ON
#define DTR ON
#define MAX_TIME 30 /* MAX_TIME secs to connect or program w
ill
terminate */
#define SECONDS 4 /* to wait for hm command to work */

```

```
int CREDIT_PORT = COM8 ;
```

```
/* 0,1 com1 and com2 etc.. */
```


CREDIT.C

```

int CREDIT_PARITY      = P_EVEN;          /* cdc req,  P_NONE, P_ODD
etc.. */
int CREDIT_STOP_BITS   = 1;              /* cdc req, 1 or 2 otherwi
se */
int CREDIT_WORD_LENGTH = 7;              /* cdc req, 7 data bits, 1
parity */
int CREDIT_BAUD         = 1200;          /* cdc req baud */
int CREDIT_DUPLEX       = FULL;          /* cdc req*/

/* variables that change are now listed */
clock_t clock(void);
int TIME;                                /* used for timing out the transaction,
timing it */
int PORT_ERROR;                          /* return value for port errors */
char far authnumber[80];
int error_check;
char far cdc_response[255];              /* a generic response to be
displayed */
int far response_code;                   /* a code generated to give status
of
transaction */
int far program_error;                   /* returns <0 if there was a program
exec probelm */
char far authorization_number[80]; /* authorization number for approva
l */

char far callnumber [80];                /* phone number to call for person to
person authorization*/

char far card_name [80];
char far card_number[255];               /* card_number storage*/
char far expr_date[255];                 /* expiration data */
char far approval_amt[255];              /* approval amount */
char far code[255];                     /* transaction code */
char far siteid[15];                     /* site id */
char far phone_no[80];                   /* a phone_no to call for credit auth */

/*****
* credit_open_port
* params : takes a communication port, int port
* returns : error code
* function: opens the port for Async communications.
*****/
int credit_open_port (port)
int port;
{
int stat;

stat = ASSUCCESS;

/*
if ( (system_type == SAMSUNG) || (system_type ==

```


CREDIT.C

```

RO_SYSTEMS) )
        if ( (stat = asisetv
(port, 0x2E8, 13, 0x20, IRQ5, 2, NO, 0, 0, 0)) < ASSUCCESS) {
                return (stat);
        }
*/

        if ((stat = asifirst (port, MODE, RXLEN, TXLEN)) < ASSUCCESS){
                return (stat);
        }
        if ((stat = asiinit(
port, CREDIT_BAUD, CREDIT_PARITY, CREDIT_STOP_BITS, CREDIT_WORD_LENGTH))
                < ASSUCCESS ) {
                return (stat);
        }
        if ( (stat = asdtr(port, ON)) < ASSUCCESS)
                return stat;
        if ( (stat = asrts (port, ON)) < ASSUCCESS)
                return stat;
        if ( (stat = asistart(port, ASINOUT)) < ASSUCCESS)
                return stat;
        return (stat);
}

/*****
initialize_modem
  setup_modem
*****/
initialize_modem (port)
int port;
{
int error;
        error_check = FALSE; /* diagnostics variable*/
        use (auth_wt);
        clrscr();
        cprintf ("                Initializing Modem...");
        HMWaitForOK (TICKS_PER_SECOND*SECONDS, NULL);
        error = HMReset (port); /* reset modem */
        if (error < ASSUCCESS) {
                use (auth_wt);
                clrscr ();
                cprintf ("                ERROR: Can't Reset modem");
                return error;
        }
        if (ECHO == 0)
                if ( (error = HMSetEchoMode (port, OFF)) < ASSUCCESS)

/* set echo */
                return error;
        if (ECHO == 1)
                if ( (error = HMSetEchoMode (port, ON)) < ASSUCCESS)
                        return error;
        if ( (error = HMSetVerboseMode (port, ON)) < ASSUCCESS)

```


CREDIT.C

```

        return error;
    /* verbal response */
    if ( (error = HMSetFullDuplexMode (port,ON)) <
ASSUCCESS)/* duplex FULL */
        return error;
    if ( (error = HMSetSpeaker (port,SPEAKER)) <ASSUCCESS) /*
set speaker */
        return error;
    return (ASSUCCESS);
}

/*****
* do_dial()
* params :takes a port and a phone_number.
* returns : none.
* function: dials up host.  steps in logic
*          1 - reset modem
*          2 - set specs
*          3 - dial modem
*****/
int do_dial (port,phone_number)
int port;
char phone_number[];
{
    int error;
    char modem_cdc_response,dial_string[40];
    clock_t start,end;

    modem_cdc_response = NULL;
    if ((error = initialize_modem (port)) != ASSUCCESS) return (err
or);

        use (auth_wt);
        clrscr();
        cprintf ("                Dialing...");
        HMSetWaitTimeForCarrier (port,60);
        if ( (error = HMDial (port,phone_number)) <ASSUCCESS) /*
dial number */
            return error;

        start = clock ();
        end = clock ();
        while (!is_connected_CDC (port)) {
            modem_cdc_response = asigetc (port);
            if (modem_cdc_response == 'B'){
                return (-21);} /* BUSY
signal */

                gotoxy (40,1);
            end = clock ();
            cprintf ("%d", (int)((end-start)/CLK_TCK));
            if ( (int)((end-start)/CLK_TCK) >= MAX_TIME)
                return (-26);
        }
        clrscr ();
        cprintf ("                Connected!");

```


CREDIT.C

```

        return (ASSUCCESS);
    }

    /*******
    credit_hang_up
        disconnects the phone.
    *****/

int credit_hang_up (port)
int port;
{
    int i,stat;
        for (i=1;i<4;++i)
            asiputc (port, '+');          /* send hang up string */
        if ( (stat = send_string(port, "ATH0\r\0")) < ASSUCCESS){/* go on
hook */
            asiquit (port); /* try and shut off interrupts anywar */
            return stat;
        }
        stat = asiquit (port);             /* deativate greenleaf */
        return stat;
    }

    /*******
    is_connected_CDC
        returns true if we are is_connected_CDC to host
    *****/

int is_connected_CDC (port)
int port;
{
    int stat;
        stat = (iscd (port, CUMULATIVE));
        return stat;
    }

    /*******
    * calc_LRC ()
    *  params : char t[]
    *  returns: char
    *  function: calculates the exclusive Oring of all the char's
    *             in t with the exception of the first character.
    *****/

char calc_LRC (t)
char t[];
{
    int i;
    char LRC;
        i = 1; /* start at 1 to skip the STX char, first char */
        LRC = t[i];
        i++;

```


CREDIT.C

```

while (t[i] != '\0'){
    LRC = LRC^t[i];
    ++i;
}
return (LRC);
}

```

```

/*****

```

```

* build_transmission_string ()
*  params : takes card_number,expr_date,approval_amt,code
*  returns : (char *)
*  function: builds the data string that will be transferred to
*             cdc for authorization.
*****/

```

```

char *build_transmission_string (card_number,expr_date,
                                approval_amt,code,siteid,auth_no)

```

```

char card_number[];
char expr_date[];      /* card expiration date */
char approval_amt[];
char code[];           /* transaction code */
char siteid[];         /* site id*/
char auth_no[];
{
char transmission_string[255];
char LRC;
int i;  i = 0;
    transmission_string[i] = '\0';      /* start string with NULL
*/
    strNCat (transmission_string,2);    /* Hex 2, stx */

    /* a 'P.' spec hex 502E */
    strNCat (transmission_string,80);   /* hex 50 dec 80*/
    strNCat (transmission_string,46);   /* hex 2E dec 46*/

    strcat (transmission_string,"000000"); /* 6 zeros */
    strcat (transmission_string,siteid); /* 5 char site id */
    strcat (transmission_string,"00000"); /* 5 zeros */

    strNCat (transmission_string,28); /* hex 1C, spec FS char */

    /* WCC char configuration*/
    /* bit 7 (MSB) Even parity bit */
    /* bit 6 Always a '1' */
    /* bit 5-2 not used */
    /* bit 1 multiple transaction indicator bit */
    /* bit 0 Magnetic stripe indicator bit */
    /* currently the WCC looks like */
    /* bin 01000000 or dec 64 */
    strNCat (transmission_string,64); /* WCC */
}

```


CREDIT.C

```

    strcat (transmission_string,card_number);
    strNCat (transmission_string,28);    /* hex 1C, FS char */

    strcat (transmission_string,expr_date);

    strNCat (transmission_string,28);    /* FS char */

    strcat (transmission_string,approval_amt);

    strNCat (transmission_string,28);    /* FS char */

    strcat (transmission_string,code);
    /* code determines what type of transaction */
    /* that will occur */

    strNCat (transmission_string,28);    /* FS char */

    strcat (transmission_string,auth_no); /* auth code */
    strNCat (transmission_string,28);    /* FS char */
    strNCat (transmission_string,3);    /* ETX char hex 03 */

    LRC = calc_LRC (transmission_string);

    strCHcat (transmission_string,LRC);

    strCHcat (transmission_string,'\0');
    return (transmission_string);
}

```

```

/*****

```

```

    Send 4 periods to allow CDC to detect our baud rate.
*****/

```

```

send_baud_detect (port)
int port;
{
    int stat;

```

```

    delay (4000); /* wait exactly 4000 ms or 4 secs */
    if (!is_connected_CDC (port)) return (-99); /* connection failed */
    if ((stat = asiputc (port, '.')) < ASSUCCESS) return stat;
    if (!is_connected_CDC (port)) return (-99); /* connection failed */
    delay (200); /* 200 ms delay */
    if ((stat = asiputc (port, '.')) < ASSUCCESS) return stat;
    if (!is_connected_CDC (port)) return (-99); /* connection failed */
    delay (200); /* 200 ms delay */
    if ((stat = asiputc (port, '.')) < ASSUCCESS) return stat;
    if ((stat = asiputc (port, '.')) < ASSUCCESS) return stat;
    if ((stat = asiputc (port, '\r')) < ASSUCCESS) return stat;
    return ASSUCCESS;
}

```

```

/*****

```


CREDIT.C

```

    send_password
    log onto EFTDS, funds transfer
    *****/

int send_password (port)
int port;
{
    int ch,error = 0;
    clock_t start,end;
    start = clock ();
    end = clock ();
    while ( ((ch = asgetc (port)) != '>') && ( ((end-start)/CLK_TCK) <
MAX_TIME) ){
        end = clock ();
    }
    if ( (end-start)/CLK_TCK >= MAX_TIME)
        return -10;
    delay (200); /* wait 200 ms before sending EFTDS */
    error = HMSendStringNoWait (port,"EFTDS","\r");
    return error;
}

/*****
* talk ()
* params : port, and a data string
* returns : none.
* function: holds the connection between host and terminal.
*           sends and recieves authorization for credit cards.
*****/

int talk (port,transmit_string)
int port;
char transmit_string[];
{
    char recieve_string[255];
    char helper[255];

    int stat,CDC_LRC,LRC;
    int i;
    int ours,error;
    int transmitting = 0;

    hmcarron (port,1); /* turn carrier on */
    use (auth_wt);
    clrscr();
    cprintf ("                Sending Baud Detect...");
    if (!is_connected_CDC (port)) return (-99); /* connection failed
*/
    if ((stat = send_baud_detect (port)) < ASSUCCESS) return (-23);
    clrscr ();
    cprintf ("                Sending Password...");
    if ((error = send_password (port)) < 0) return (error);
    if (!is_connected_CDC (port)) return (-99); /* connection failed */
    clrscr ();

```


CREDIT.C

```

cprintf ("                Sending EFT Request...");

i = stat = 0;
recieve_string[i] = '\0';

while ((stat != 4)){ /* hex 4 is EOT */
    if (!is_connected_CDC (port)) return (-99); /* connection
failed */

    while ((stat = asgetc (port)) > -1){
        switch (stat) {
            case 5 : /* ENQ they want the data*/
                                use (auth_wt);
                clrscr();
                cprintf ("                Transmitting...")
;
                transmitting = 1;
                error = send_string (port,transmit_string);
                                if (error <
ASSUCCESS) {
                                return (error);
                }
                                /* wait for buffer to empty */
                                /* means string was send */
                                if
(!is_connected_CDC (port)) return (-99); /* connection failed */
                                while ((!istxempty (port))) {
                                                                if
(!is_connected_CDC (port)) {
                                                                return (-99); /* connection failed
*/
                                                                }
                                                                }
                                break;
            case 21 : /* NAK, CDC didn't understand us */
                                use (auth_wt);
                clrscr();
                cprintf ("                Re-Trying
Transmission...");
                error = send_string (port,transmit_string);
                if (error) return (-23);
                                /* wait for buffer to empty */
                                /* means string was send */
                                if
(!is_connected_CDC (port)) return (-99);
                                while ((!istxempty (port)))
                                                                if
(!is_connected_CDC (port)) return (-99); /* connection failed */
                                break;
            case 2 : /* STX sending data to us */
                                use(auth_wt);
                clrscr();
                cprintf ("                Receiving...")
;

```


CREDIT.C

```

recieve_string[0] = '\0';
strCHcat (recieve_string,2); /* add STX */
while ((stat = asgetc (port)) != 3) { /*ET
X*/
    if (stat > -1)
        strCHcat (recieve_string,stat);
    if
(!is_connected_CDC (port)) return (-99); /* connection failed */
    }
    /* add ETX to string for calculating LRC */
    strcpy (helper,recieve_string);
    strNUMcat (helper,3);
    while ((stat = asgetc (port)) <= -1)
        if
(!is_connected_CDC (port)) return (-99); /* connection failed */
    CDC_LRC = stat;
    LRC = calc_LRC (helper);
    /* this next if determines if we understood
CDC or not */
    if (LRC == CDC_LRC){
        asputc (port,6); /* ACK char */
    }else {
        asputc (port,21); /* NAK char*/
    }
    /* zap ETX from end of
string */
zap_ETX (recieve_string);
    if (!is_connected_CDC (port)) return (-99);
    /*
connection failed */
    /****** temp until find out about eot */
goto done;
        break;
    case 4 : /* EOT */
        if
(!is_connected_CDC (port)) return (-99); /* connection failed */
        goto done;
        break;
    }
}
done: strcpy (cdc_response,recieve_string);
return (0); /* success */
}

/* puts a NULL on the end of a string over the ETX char */
zap_ETX (s)
char s[];

```


CREDIT.C

```
{
int add; /* address positioning */
    add = 0;
    while (s[add] != 3) ++add;
    s[add] = NULL;
}
```

send a command to the modem

*****/

sendToModem (port,t)

int port;

char *t;

{

int stat,error;

error = FALSE;

while (*t){

if ((stat = asiputc (port,*t++)) < ASSUCCESS)

return (stat);

}

stat = asiputc (port,'\r');

return (stat);

}

* send_string (port,t) port, and string t

* returns : error, -23, if can't place char in buffer

* functions: sends a string through the modem

*****/

int send_string (port,t)

int port;

char *t;

{

int stat;

while (*t){

if ((stat = asiputc (port,*t++)) < ASSUCCESS)

return (stat); /* put error to port */

}

return (ASSUCCESS);

}

based on format they send back

pray it doesn't change

*****/

extract_authnumber (val,r,a)

CREDIT.C

```

int val;
char r[],a[];
{
    int i = 0;
    switch (val) {
        case 1:
            *a = NULL;
            while (*r != ' ') ++r; /* get passed word
"approval" */
            while (*r == ' ') ++r; /* move to start of auth
code */
            while (i <= 6) { /* first 6 digits are the auth
code */
                *a++ = *r++;
                ++i;
            }
            *--a = NULL;
            break;
        case 2:
            *a = NULL;
            while (*r != ' ') ++r; /* get passed word
"accepted" */
            while (*r == ' ') ++r; /* move to start of batch
code */
            while (i <= 8) { /* first 8 digits are the
batch code */
                if (*r != '-') { /* take out the '-'
saves 1 digit of spac*/
                    *a++ = *r++;
                } else r++;
                ++i;
            }
            /* don't add null because field is only 8 chars,
shit */
            break;
    }
}

```

```

/** getToken gets the next word from a string ending with a space **/
/** and stores the word in t **/
getToken (s,t)
char s[],t[];
{
    int add;
    add = 0;
    while (((t[add] = s[add]) != ' ') && (t[add] != NULL)) ++add; /*
get
a word ending with space */
    t[add] = NULL;
}

```

parse_cdc_response (r)

CREDIT.C

```

char r[];
{
char a[80],error[80];
int parsed; parsed = 0;
    *a = NULL;
    getToken (++r,a);    /* skip STX and get first word from r into a */
    if (strcmp (a,"DECLINED",8) == 0) {
        strcpy (cdc_response,"Declined - Please get another card.");
        response_code = -1;
        parsed = 1;
    }
    if (strcmp (a,"APPROVAL",8) == 0) {
        extract_authnumber (1,r,authnumber);
        strcpy (cdc_response,"Approved - Authorization Number :
");
        strcat (cdc_response,authnumber);
        strcpy (authorization_number,authnumber);
        response_code = 0;
        CARD_APPROVED = TRUE;
        parsed = 1;
    }
    if (strcmp (a,"ACCEPTED",8) == 0) {
        extract_authnumber (2,r,authnumber);
        strcpy (cdc_response,"Approved - Batch and Item Number :
");
        strcat (cdc_response,authnumber);
        moveX (authorization_number,authnumber,8);
        /* do a moveX so as not to overright next field in
agreemntrec */
        /* This is because of space problems in the
current agreemtnrec */
        response_code = 0;
        CARD_APPROVED = TRUE;
        parsed = 1;
    }
    if (strcmp (a,"HOLD-CALL",4) == 0) {
        strcpy (cdc_response,"Confiscate card - (if safe) and
use another");
        response_code = -2;
        parsed = 1;
    }
    if (strcmp (a,"CALL",4) == 0) {
        *callnumber = NULL;
        strcpy (cdc_response,"Call - ");
        extract_phonenumber (r,callnumber);
        strcat (cdc_response,callnumber);
        response_code = -3;
        parsed = 1;
    }
    if (strcmp (a,"INVALID",7) == 0){
        *error = NULL;
        strcpy (cdc_response,"An ");
        strcat (cdc_response,r);
        strcat (cdc_response," was entered. Try again.");
    }
}

```


CREDIT.C

```

        response_code = -4;
        parsed = 1;
    }
    if (strncmp (a,"UNAVAILABLE",11) == 0) {
        strcpy (cdc_response,"Draft capture not available.");
        response_code = -5;
        parsed = 1;
    }
    if (strncmp (a,"RE-ENTER",8) == 0) {
        strcpy (cdc_response,"Credit card authorization failure.
Try again.");
        response_code = -6;
        parsed = 1;
    }
    if (strncmp (a,"NON-SUBSCRIBER",3) == 0) {
        strcpy (cdc_response,"We do not subscribe to requested
credit card.");
        response_code = -7;
        parsed = 1;
    }
    if (strncmp (a,"AGENCY",6) == 0) {
        strcpy (cdc_response,"Transmission failed. No response.
Try again.");
        response_code = -8;
        parsed = 1;
    }
    if (parsed != 1) {
        strcpy (cdc_response,r);
        set_error (-20);
        response_code = -20;
    }
}
}

```

```

/* extract phone number from cdc's cdc_response */
extract_phonenumber (r,p)
char r[],p[];
{
    int i;
    r = r + 4; /* skip the word CALL 4 chars long */
    i = 0;
    while ((p[i] = r[i]) && (i <= 11)) ++i;
}

```

```

/* main procedure logic for cdc specs */

```

```

int call_cdc
(card_number,expr_date,approval_amt,code,siteid,phone_no,auth_no)
char card_number[]; /* card_number storage*/
char expr_date[]; /* expiration data */
char approval_amt[]; /* approval amount */

```


CREDIT.C

```

char code[];          /* transaction code */
char siteid[];        /* siteid */
char phone_no[];      /* phone number to call */
char auth_no[];
{
int stat,error;
int approval_code;
char transmit_string[255];

CARD_APPROVED = FALSE;
strcpy (transmit_string,build_transmission_string (card_number,
                                                    expr_date,
                                                    approval_amt,
                                                    code,siteid,auth_no));
strcpy (cdc_response,"*- Unknown System Error -* Call TELEMAT");
response_code = -1; /* set to declined */
program_error = 0; /* set to none */
if ((PORT_ERROR = credit_open_port(CREDIT_PORT)) != ASSUCCESS){
    program_error = PORT_ERROR;
    set_error (PORT_ERROR);
    return (PORT_ERROR);
}
error = do_dial (CREDIT_PORT,phone_no);
if (error == -21){
    program_error = -21;
    set_error (-21);
    response_code = -21;
    return (-21);
}
if (error == -99){ /* connection broken */
    program_error = -99;
    set_error (-99);
    response_code = -99;
    return (-99);
}
if (error == -26) { /* no phone lines, can't connect
*/
    program_error = -26;
    set_error (-26);
    response_code = -26;
    return (-26);
}
if (error != ASSUCCESS) {
    set_error (error);
    return (error);
}
error = talk (CREDIT_PORT,transmit_string);
if (error == -23){
    program_error = -23;
    set_error (-23);
    response_code = - 23;
    return (-23);
}
if (error == -99){

```


66

CREDIT.C

```

        program_error = -99;
        set_error (-99);
        response_code = -99;
        return (-99);
    }

    parse_cdc_response (cdc_response);
    return (error);
}

char *print_TF (n)
int n;
{
    if (n){ return ("YES");}
    else return ("NO");
}

set_error (error)
int error;
{
    switch (error ) {
        case -1 : strcpy (cdc_response, "Not Otherwise Defined
Error!");
                    break;
        case -2 : strcpy (cdc_response, "Requested Port Out of
Range!");
                    break;
        case -3 : strcpy (cdc_response, "Port Already Setup!");
                    break;
        case -4 : strcpy (cdc_response, "Invalid Buffer Size
Requested!");
                    break;
        case -5 : strcpy (cdc_response, "No Memory Available for
Buffer(s)!");
                    break;
        case -6 : strcpy (cdc_response, "GreenLeaf setup not run,
asiopen!");
                    break;
        case -7 : strcpy (cdc_response, "Invalid Parameter!");
                    break;
        case -10 :strcpy (cdc_response, "Function timed out!");
                    break;
        case -14 :strcpy (cdc_response, "No 8250 UART at I/O
Address. CALL TELEMAR");
                    break;
        case -20 : strcpy (cdc_response, " - Try again.");
                    break;
        case -21 : strcpy (cdc_response, "Phone was Busy. Try
again.");
    }
}

```


CREDIT.C

```

                                break;
        case -22 : strcpy (cdc_response,"Modem NOT Responding...");
                                break;
        case -99 : strcpy (cdc_response,"Connection broken.  TRY
AGAIN!.");
                                break;
        case -23 : strcpy (cdc_response,"Error in data
Transmission.      Try again.");
                                break;
        case -24 : strcpy (cdc_response,"--- Disk Error
---(ira004.net)  CALL TELEMAC");
                                break;
        case -25 : strcpy (cdc_response,"--- Disk Error
---(ira004.dat)  CALL TELEMAC");
                                break;
        case -26 : strcpy (cdc_response,"No answer or No phone
line cables!");
                                break;
        case -27 : strcpy (cdc_response,"Computer's clock has
malfunctioned.");
                                break;
    }
}

```

```

int check_for_errors (port)
int port;
{
    int stat;
    if (error_check){
        printf ("                Communications Diagnostics\n");
        stat = isalert (port); printf ("\nNOTE - Is alert flag set?
%s",print_TF(stat));
        stat = isrxempty (port); printf ("\nNOTE - Is RX Buffer
Empty? %s",print_TF(stat));
        stat = isrxoverflow (port); printf ("\nNOTE - Is RX Buffer
Overflow? %s",print_TF(stat));
        stat = istxempty (port); printf ("\nNOTE - Is TX BUFFER empty?
%s",print_TF(stat));
        stat = islinerr (port); printf ("\nNOTE - Is line error?
%s",print_TF(stat));
        stat = ismodemerr (port); printf ("\nNOTE - Is modem error
checking? %s",print_TF(stat));
        stat = istxintrunning (port); printf ("\nNOTE - Are tx interrup
pts
running? %s",print_TF(stat));
        stat = isrxintrunning (port); printf ("\nNOTE - Are rx interrup
ts
running? %s",print_TF(stat));
        stat = isigalert (port); printf ("\nNOTE - Is alert being igno
red
? %s",print_TF(stat));
    }
}

```


CREDIT.C

```

stat = isigcts (port); printf ("\nNOTE - Is CTS being ignored?
%s",print_TF(stat));
stat = isigdsr (port ); printf ("\nNOTE - Is DSR being
ignored? %s",print_TF(stat));
stat = isigcd (port ); printf ("\nNOTE - Is CD being ignored
%s",print_TF(stat));
stat = isigmstat (port ); printf ("\nNOTE - Are modem status
changes being ingored? %s",print_TF(stat));
stat = isiglstat (port ); printf ("\nNOTE - Are receiver error
s
being ingnored? %s",print_TF(stat));
stat = isoverrun (port,DIRECT); printf ("\nNOTE - Has a Receiv
er
overrun Error occured? %s",print_TF(stat));
stat = isparityerr (port,DIRECT); printf ("\nNOTE - Has a
Parity error occured? %s",print_TF(stat));
stat = isframerr (port,DIRECT); printf ("\nNOTE - Has a Framin
g
error occured? %s",print_TF(stat));
stat = isbreak (port,DIRECT); printf ("\nNOTE - Has a
Break signal been received? %s",print_TF(stat));
stat = isxoffblocked (port); printf ("\nNOTE -
Transmitter blocked due to XOFF? %s",print_TF(stat));
stat = isctsblocked (port); printf ("\nNOTE - Transmitter
blocked due to CTS not asserted? %s",print_TF(stat));
stat = is_connected_CDC (port); printf ("\nConnect Status
- State of Carrier Detect, (is_connected_CDC to host) ? %s",print_TF(st
at));
printf ("\n\n");
}
}

do_time ()
{
Lock();
if ((TIME = clock()) == (clock_t)-1){
program_error = -27;
response_code = -27;
set_error (-27);
Unlock();
return (-27);
}
if (TIME/CLK_TCK > MAX_TIME) {
program_error = -26;
response_code = -26;
set_error (-26);
Unlock();
return (-26);
}
Unlock();
}

```



```

delay (0);
if (trans_code[0] == '\0') strcpy (trans_code, " ");
    CREDIT_PORT = port;
    auth_wt = wt;
    use (auth_wt);
    _setcursortype (_NOCURS);

    sprintf (s, "Credit Card Authorization (%d)", CREDIT_PORT+1);
    setttitle (auth_wt, s, CenterUpperTitle);
    stat = 1;
    strcpy (authorization_number, "DECLINED");

    strcpy (card_number, credit_number);
    null_end (card_number, 19); /* put null after last
character*/

    strcpy (expr_date, expr);
    sprintf (approval_amt, "%1.2f", appamt);
    strcpy (siteid, site);
    strcpy (code, trans_code);
    strcpy (phone_no, phone);

    TIME = 0;
    stat = call_cdc
(card_number, expr_date, approval_amt, code, siteid, phone_no, auth_number);
    use (auth_wt);
    clrscr();
    cprintf ("                Hanging up phone...");

```


CREDIT.C

```

strcpy (response,cdc_response);
_setcursortype (_NORMALCURSOR);
if ( (stat1 = credit_hang_up (CREDIT_PORT)) < ASSUCCESS) {
    clrscr ();
    cprintf ("          asiquit : Interrupt error
%d",stat1);
}
if (CARD_APPROVED) {
    update_tau_status (4,'9');
    use (auth_wt);
    clrscr();
    cprintf ("          APPROVED
%s",authorization_number);
    /* moveX because null overrides next
field in agreemtnrec*/
    /* this is temporary until we change
agreemntrec */
    moveX (auth_number,authorization_number,8);
    strcpy (response,cdc_response);
    return TRUE;
} else
if ( (stat == 0) && (!CARD_APPROVED) ) {
    use (auth_wt);
    clrscr ();
    cprintf ("%s",cdc_response);
    strcpy (response,cdc_response);
    return FALSE;
} else {
    if (response_code == -1) {
        update_tau_status (4,'8');
    } else {
        update_tau_status (4,'7');
        use (auth_wt);
        clrscr();
        cprintf ("Error: %s",cdc_response);
        strcpy (response,cdc_response);
        return FALSE;
    }
}
}

```


CREDIT.C

```

null_end (char *s,int l)
{
int i,j;
    for (i=0;i<l;i++)
        if ( (s[i] != ' ') && (s[i] != '\0') )
            j = i;
    s[j+1] = '\0';
}

int get_credit (float appamt,
                wintype wt,
                char *credit_number,
                char *expr,
                char *site,
                char *phone,
                char *trans_code,
                char *response,
                char *auth_number,
                int port)
{
int stat,stat1;
char amount[20];
char s[80];

```


CCOPYIT.C

```

/*-----
MODULE: ccopyit

Description: To a file from hard disk to floppy; requiring that
            the floppy disk be present

Entry Function: main
Exit Function: main

Written By : Greg McGregor

Revisions:
GMM 9-6-1991    copies to b: drive now instead of a:
-----*/

```

```
/* includes */
```

```

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <io.h>
#include <bios.h>

```

```
#include <windows.h> /* my windows package */
```

```

#define FALSE 0
#define TRUE 1

```

```

windef info_win = {10,10,70,13,White,Red,FALSE,FALSE,FALSE,TRUE,SINGLEFR
AME,
                White,Red};
wintype info_wt;

```

```

/*-----
*
* Procedure Name: main
* Parameters:
* Function:
* Returns:
*
* Written By: Greg McGregor
-----
-*/

```

```

void main (int argc,char *argv[])
{
char command[80];

if (argc != 2) {
    clrscr ();

```


CCOPYIT.C

```

printf ("\n\nccopyit V1.01");
printf ("\nUSAGE> ccopyit <source>");
printf ("\n\nccopyit: Copies the source file onto drive b:");
printf ("\n\nGMM 1991");
exit (0);
} else {
    while (!check_for_destination_disk ()) {
        info_wt = windowopen (&info_win);
        setttitle (info_wt,"ERROR",CenterUpperTitle);
        use (info_wt);

        clrscr ();
        cprintf ("                Please Place insert a floppy disk!"
);
        gotoxy (1,2);
        cprintf ("                Press <ESC> Key");
        getch ();
        windowclose (info_wt);
    }
}
sprintf (command,"copy %s b:\%s > out",argv[1],argv[1]);
system (command);
exit (0);
}

```

```

/*-----
*
* Procedure Name: check_for_destination_disk
* Parameters:
* Function:
* Returns: True, FALSE
*
* Written By: Greg McGregor
-----

```

```

-*/
int check_for_destination_disk () {
int stat;
char buff[2048];
    stat = biosdisk (2,1,0,1,1,1,buff);
    if (stat == 0) return TRUE;
    stat = biosdisk (2,1,0,1,1,1,buff);
    if (stat == 0) return TRUE;
    return FALSE;
}

```


CARDRDR.C

```
/*-----  
-----
```

MODULE: cardrdr.c works with MAGTEK card readers

Credit card reader source file.
MAGTEK

Written By : Greg McGregor

REVISION:	What was revised?
- GMM 7-30-1991	Nothing

```
-----  
---*/
```

```
#include <stdio.h>  
#include <string.h>  
#include <gkeys.h>  
#include <time.h>  
#include <windows.h>  
#include <misc.h>
```

```
int tracks_read = 0;  
int MAX_TRACKS = 2;  
int READER_TIME_OUT = 10; /* time out */
```

```
/*  
//  
// get_reader_char  
//  
*/  
int get_reader_char (wintype wt) {  
    clock_t start,current;  
    int x,y,osc = 0;  
        x = wherex ();  
        ++x;  
        y = wherey ();  
        start = clock ();  
        current = clock ();  
        while ( (!kbhit ()) && ( (current-start)/CLK_TCK <  
READER_TIME_OUT) ) {  
            current = clock ();
```


CARDRDR.C

```

    }
    if ( (current-start)/CLK_TCK >= READER_TIME_OUT) return ( 0xFFFF
);
    return ( (int)getch ( ) );
}

```

```

int read_in_card (char *card_number,

```

```

                    char *card_name,
                    char *card_expr,
                    char *card_type) {

```

```

int i;
char c;
int ci;
char tmp[255];
char track[4][255]; /* allows up to 4 tracks, 255 char long to be
read */
wintype wait_wt;

```

```

    wait_wt = wait_window (" * Reading Creditcard Information *")
    tracks_read = 0;
    while (tracks_read < MAX_TRACKS){
        *track[tracks_read+1] = NULL; /* 13 return */
        while ( ci = get_reader_char (wait_wt) ) {

```

```

            if ( ci == 0xFFFF ) {
                windowclose (wait_wt) ;
                return ( FALSE );
            }

```

```

            c = (char)ci;
            strCHcat (track[tracks_read+1],c);
            /* if user mistakenly hits a
            /* char used by the card read
            /* by typing a RETURN or an

```

```

which is the first*/

```

```

They can exit */

```

```

ESCAPE */

```

```

            if ((c == 0x0D) || (c == 0x1B)) {
                windowclose (wait_wt);
                return ( FALSE );
            }

```

```

    }

```


CARDRDR.C

```
        ++tracks_read;
    }
    ci = get_reader_char (wait_wt);
    windowclose (wait_wt);
    if ( ci == 0xFFFF ) return ( FALSE );
        /* get the return, dec 13, at end of string */

    strcpy (tmp,track[1]);
    extract_account (track[2],card_number,card_type);

    extract_name (tmp,card_name);

    format_name (card_name);

    extract_expr (track[2],card_expr);

    return ( TRUE );
}
```

```
extract_account (t,card_number,card_type)
char t[];
char *card_number;
char *card_type;
{
    int add = 0;
    int add1 = 0;

    *card_number = NULL;
    if (t[add] == '%') add += 2;    /* put pointer to start of
account
number*/
    else ++add; /* assume at char after % */

    while ((card_number[add1++] = t[add++]) != '=') ; /* track
2 */
    card_number[--add1] = NULL; /* erase the ~ from account n
umber */

    switch (card_number[0]){
```


CARDRDR.C

```

        case '3' : if (card_number[1] == '7')
                    strcpy (card_type, "AE");
                    if (card_number[1] == '8')
                        strcpy
(card_type, "DC"); /* diners club */
                    break;
        case '4' : strcpy (card_type, "VI");
                    break;
        case '5' : strcpy (card_type, "MC");
                    break;
        case '6' : strcpy (card_type, "DI"); /* discov
*/
                    break;
        case '9' : strcpy (card_type, "CB");
                    break;
    }
}

```

```

extract_name (t, card_name)
char t[];
char *card_name;
{
    int add = 0;
    int add1 = 0;

    *card_name = NULL;

    while (t[add++] != '^') ; /* find start of name, field se
perated
by ^ */

    /* end of name end with ^ also */

    while ((card_name[add1] = t[add]) != '^') {
        ++add1;
        ++add;
    }
    /* left in the ^ and the end of the string */
    /* for use in format_name */
    card_name[++add1] = NULL;
}

```


CARDHDR.C

```

format_name (char *card_name)
{
    int add, add1;
    char tmp[255];
        add = add1 = 0;
        tmp[0] = NULL;
        while ((card_name[++add]) != '/') ;
        while ((tmp[add1] = card_name[++add]) != '^') ++add1;
        add = 0;
            /* add a space between middle initial and last nam
e*/
            /* if middle initial doesn't exist a space will be
*/
            /* between first and last name */
        tmp[add1] = ' ';
        ++add1;
        while ((tmp[add1] = card_name[add]) != '/') {
            ++add1;
            ++add;
        }
        tmp[add1] = NULL;

        cut_out_Xspaces (tmp);

        strcpy (card_name, tmp);
}

```

```

cut_out_Xspaces (t)
char t[];
{
    char tmp[50];
    int add = 0;
    int add1 = 0;
        while ((tmp[add] = t[add1]) != ' ') {
            ++add;

```


CARDRDR.C

++add1;

)


```

while (t[add1] == ' ') ++add1;
++add;
while ((tmp[add] = t[add1]) != NULL){
    ++add;
    ++add1;
}
strcpy (t,tmp);
}

```

```

extract_expr (t,card_expr)
char t[];
char *card_expr;
{
int tmp[5];
int add,add1,i;
    add = add1 = 0;
    *tmp = NULL;
    while ((t[add] != '=')) ++add;
    for (i=1;i<=4;++i)
        tmp[add1++] = t[add];
    tmp[add1] = NULL;

```

```

        card_expr[0] = tmp[2];
        card_expr[1] = tmp[3];
        card_expr[2] = tmp[0];
        card_expr[3] = tmp[1];
        card_expr[4] = NULL;

```

```

}

```

```

/*
main ()
{
char c;
    while (((c = getch ()) != '%')) ;
    read_in_card ();
    printf ("\nCard type : %s",card_type);

```


CARDRDR.C

```
printf ("\nName is : '%s'",card_name);  
printf ("\nAccount number : '%s'",card_number);  
printf ("\nExpr date is: '%s'",card_date);
```

```
}  
*/
```


ARCHIVE.C

```
/*-----  
-  
Greg McGregor : November 5 1990  
Archive : V1.0 Switch option Overwrite or Just add  
Archives Version 2.00 & 1.90 + record total 936  
bytes  
  
Parameters: NONE  
  
Returns: NONE  
  
Side Effects :  
- imports data from the file 'agreemnt' - a sequential  
C file into the file 'archive.a'  
- Deletes closed agreements from agreemnt.  
- Squashes agreemnt file.  
  
Notes: If a file exists in the data base agreemnt and also in t  
he  
file agrb the file in the agrb will be overwritten if netdue>0  
  
MODIFIED: 7-2-1991 to version 3.0+ compatibility  
MODIFIED: 8-10-1991 with new agreement structure GMM  
-----  
*/  
  
#include <stdio.h>  
#include <rtio.h>  
#include <io.h>  
#include <fcntl.h>  
#include <alloc.h>  
#include <sys\stat.h>  
#include <proc.io>  
#include <bench.h>  
#include <agreev3.h1>  
  
/* file structs and file descriptors  
* agreemnt is defined in agreemnt.h1  
*/  
struct agreemnt_def agrbrec;  
int fd_agrb;
```


83

ARCHIVE.C

```
int keypos_agrb[16];
```

```
char in_yet0[88];
```

```
char site [80];
```

```
int flag;
```

```
/*
```

```
 * GLOBAL DEFINES
```

```
*/
```

```
#define FIDS_agrb 82
```

```
#define IMPORT_FROM "agreemnt"
```

```
#define IMPORT_TO "archive.agr"
```

```
#define HELP_FILE ""
```

```
#define SYSTEM ""
```

```
#define NEXT_PROGRAM ""
```

```
#define TOTALKEYS 3
```

```
#define PCHAR ' _'
```

```
#define FILE1 IMPORT_FROM
```

```
#define FSIZE1 sizeof(agreemntrec)
```

```
#define FILE2 IMPORT_TO
```

```
#define FSIZE2 sizeof(agreemntrec)
```

```
#define TRUE 1
```

```
#define FALSE 0
```

```
main (argc,argv)
```

```
int argc;
```

```
char *argv[];
```

```
{
```

```
int i = TRUE;
```

```
if (argc != 2) {
```

```
clrscr ();
```

```
printf ("\nArchive V2.01");
```

```
printf ("\n *- Version 3.0+ Compatible");
```

```
printf ("\n\nUSAGE: archive <SWITCH>");
```

```
printf ("\n\n\t<SWITCH> - o = overwrite, a = add only");
```



```

        printf ("\n\nnote: ");
        printf ("\n          - Archive imports agreemnt to
archive.agr!");
        printf ("\n          - Archive Deletes close agreements in
agreemnt!");
        printf ("\n          - Archive Squashes agreemnt file size!
        printf ("\n\n\nGMM 1991");
        exit (0);
    }

```

```

flag = 1;
if (argv[1][0] == 'a') {
    flag = 1;
} else
    if (argv[1][0] == 'o') {
        flag = 2;
    } else {
        printf ("\n Illegal Switch!");
        exit (0);
    }

    init ();
    strcpy (site,argv[1]);
    if ((i = import_file ()) == FALSE) import_error ();
    squash_file ();
}

```

```

squash_file ()
{
    int fd,fd1;
    char byte;

```

```

        system ("copy agreemnt agreemnt.bak > out"); /* in case of squa
h
error */
        fd = open ("agreemnt.",O_BINARY | O_RDWR,S_IREAD|S_IWRITE);
        fd1 = open ("agreemnt.tmp",O_BINARY | O_TRUNC|O_CREAT,S_IWRITE);
        if ( (fd == -1) || (fd1 == -1) ) {
            close (fd);
            close (fd1);
            exit (0);
        }
    }

```


ARCHIVE.C

```

while (read (fd,&byte,1)) {
    if (byte != '~') {
        write (fd1,&byte,1);
    } else {
        byte = ' ';
        while (byte == ' ') /* squash out deleted reco
*/
            read (fd,&byte,1);
    }
}
close (fd);
close (fd1);
}

init () {

#include <\h2\hdr\agreev3.h2>

}

int open_sequential ()
{
    if ((fd_agreemnt = open_file9(FILE1, FSIZE1, UPDATE_MODE,
keypos_agreemnt, FLDS_agreemnt, agreemnt_fld)) < IOGOOD)
        io_error9(SYSTEM, NEXT_PROGRAM);

    return (IOGOOD);
}

int close_sequential ()
{
    close_file9(fd_agreemnt);
}

/* btrieve is temporarily set to sequential until
we network it, then import will be for btrieve and so will
all the reports
*/
int open_btrieve ()
{

```


ARCHIVE.C

```

    if ((fd_agrb = open_file9(FILE2, FSIZE2, UPDATE_MODE, keypos_ag
reemnt,
FLDS_agreemnt, agreemnt_fld)) < IOGOOD)
        io_error9(SYSTEM, NEXT_PROGRAM);

    return (IOGOOD);
}

```

```

int close_btrieve ()
{
    close_file9(fd_agrb);
}

```

```

int import_file ()
{
    int stat;
    int keynum, keymatch, keynumB;
    struct agreemnt_def temprec; /* use to store a copy of the
                                record in use */
    long len;
    int fd;
    FILE *fp;

    stat = open_sequential ();
    stat = open_btrieve ();

    selectinx9(fd_agreemnt, keynum);
    stat = reset_file9 (fd_agreemnt, &agreemntrec);
    if (stat >= 0)
    do {
        temprec = agreemntrec;
        stat = exactkey9 (fd_agrb, &agreemntrec); /* gets new
agreemntrec */
        if (stat != IOGOOD ) {
            stat = addrec9 (fd_agrb, &agreemntrec);
            if (stat == IOGOOD) {
                if (agreemntrec.netdue != 0) delrec9
(fd_agreemnt);
            }
        } else {

```


87 ARCHIVE.C

```

closed agreemnt */
        if (temprec.netdue != 0) { /* update only if
            if (agreemntrec.netdue == 0)
                stat = updrec9 (fd_agrb,&tempre
/* use old agreemntrec */
            if (flag == 2) {
                stat = updrec9 (fd_agrb,&tempre
/* use old agreemntrec */
                if (stat != IOGOOD) {
                    printf ("%s Not
ARCHIVED!",temprec.agreeno);
                }
            }
            if (stat == IOGOOD) {
                delrec9 (fd_agreemnt);
            }
        }
    }
} while ( (stat = nextkey9 (fd_agreemnt,&agreemntrec)) >= 0);

stat = close_sequential ();
stat = close_btrieve ();

}

import_error()
{
}

```


ARCHIVE.C

PHONSTAT.C

```

/*-----
MODULE: phonstat.c

Description:

Entry Function:
Exit Function:

Written By : Greg McGregor

Revisions:
-----*/

```

```

#include <stdio.h>
#include <windows.h>
#include <misc.h>

```

```

/*-----
*
* Procedure Name: display_phone_status_message
* Parameters:
* Function:
* Returns:
*
* Written By: Greg McGregor
-----*/

int display_phone_status_message (char code, char *phone_number) {
char s[80];
    switch (code) {
        case '9':
            sprintf (s, "This Iphone, %s, is unregistered at this
site", phone_number);
            errrtn (s);
            return -1;
        case '0':
            sprintf (s, "This Iphone, %s, is currently IN", phone_number)
;
            errrtn (s);
            return 0;
        case '1':
            sprintf (s, "This Iphone, %s, is currently OUT", phone_number
);
            errrtn (s);
            return 1;
        case '2':
            sprintf (s, "This Iphone, %s, was reported LOST", phone_numbe
r);
            errrtn (s);
            return 2;
        case '3':

```


PHONSTAT.C

90

```
    sprintf (s, "This Iphone, %s, was reported BROKEN", phone_num  
ber);  
    errrtn (s);  
    return 3;  
}  
return 1;  
}
```


PICKLIST.C

```

/*
//
// These are picklist routines
//
// picklist.c
//
// Written By : Greg McGregor
//
//
*/

#include <stdio.h>
#include <\datawin\dw.h>

/*
// pick payment type
//
*/
int pick_payment_type () {
    int return_value;
    LISTITEM *ls;
    HWND win;
    int stat;
    char *s;
    char bucket[80*25*2];
    int curpos;

    stat = gettext (0,0,80,25,bucket);
    if (!stat) return ( FALSE );

    /*
    win = vopen (7,40,MARK,REVMARK,FRSINGLE,"After Selection, Press
    Any Key");
    vlocate (win,6,20);
    */
    ls = initlist ();
    if (adtolist (ls,"Cash Payment") < 0) return_value = FALSE ;
    if (adtolist (ls,"Check Payment") < 0) return_value = FALSE ;
    if (adtolist (ls,"NO Payment") < 0) return_value = FALSE ;
    s = listsel (5,5,5,NORML, REVNORML, "Pick List",REVNORML, FRDOUBLE,
    REVNORML, ls, (int) NULLF);
    /*
    vdelete (win , NONE);
    */
    freelist (ls,0);

    if (strcmp (s,"Cash Payment") == 0) {
        return_value = 1 ;
    }
}

```


92-

PICKLIST.C

```

} else
if (strcmp (s,"Check Payment") == 0) {
    return_value = 2 ;
} else
if (strcmp (s,"NO Payment") == 0) {
    return_value = 3 ;
} else return_value = FALSE;

stat = puttext (0,0,80,25,bucket);
if (!stat) return ( FALSE );

return ( return_value );
}

```


PRINTER.C

```

/*-----
--
MODULE: printer.c

PURPOSE: to print on the p.o.s printer, STAR

Written By :Greg McGregor

REVISED:          What was revised?
GMM 7-30-1991      Nothing
-----
-*/

#include <stdio.h>
#include <stdlib.h>
#include <bios.h>
#include <gkeys.h>
#include <time.h>
#include <bench.h>
#include <proc.io>
#include <gbase.h>
#include <agriio.h>
#include <real004.h>
#include <extnvar.h>
#include <agreev3.h>
#include <control.h>
#include <taustat.h>

extern int fd_real004;

#define LPT_PORT 0    /* LPT1 = 0 and so on.. */

/* check HIGH BYTE?? */
#define PRT_NOT_BUSY      0x80      /* bit 7 */
#define PRT_ACKNOWLEDGE  0x40      /* bit 6 */
#define PRT_PAPER        0x20      /* bit 5 */
#define PRT_SELECTED     0x10      /* bit 4 */
#define PRT_IO_ERROR     0x08      /* bit 3 */
#define PRT_TIME_OUT     0x01      /* bit 0 */

static FILE far *prt_fp;
int current_printer_position; /* for tab use */
int line_count;

/*
 * Put a null at end of string before any trailing spaces
 */
zap_trailing_spaces (char *s)
{
    int i,pos;

```


PRINTER.C

```

    i = pos = 0;
    while (s[i]) {
        if (s[i] != ' ')
            pos = i;
        ++i;
    }
    s[pos+1] = '\0';
}

print_tab (stop_at)
int stop_at;
{
    int i;
    while (current_printer_position < stop_at){
        ++current_printer_position;
        fprintf (prt_fp, " ");
    }
    current_printer_position = stop_at;
}

print_newline (i)
int i;
{
    int l;
    for (l=1; l<=i; ++l)
        fprintf (prt_fp, "\n");
    current_printer_position = 1;
    ++line_count;
}

/*****/
/* since some of the fields don't end with an end-of-string
   character, the field length has to be specified. Therefore
   print_section is used */
/*****/

int print_section (s,i)
char *s;
int i;
{
    int l;
    current_printer_position += i;
    for (l=0; l<i; ++l){
        if (s[l]==NULL) {
            fprintf (prt_fp, " ");
        } else fprintf (prt_fp, "%c", s[l]);
    }
}

```


PRINTER.C

```

print_string (s)
char *s;
{
current_printer_position += strlen (s);
    fprintf (prt_fp,"%s",s);
}

int print_X ()
{
    fprintf (prt_fp,"X");
    ++current_printer_position;
}

/*-----
it prints the current agreement that is
in memory at the time it is called.
-----*/
print_contract (int program_number,int lost_phone){
int stat;

    unsigned status;
    unsigned data = 0;
    status = biosprint (2,data,LPT_PORT);

    if (!(status & PRT_NOT_BUSY) && (status & PRT_PAPER) ) {
        prt_error_number = -1;
        prt_error_message [0] = '\0';
        strcat (prt_error_message,"Printer Error - OUT OF
PAPER.");
        update_tau_status (2,'5');
        return;
    }

    if ( !(status & PRT_NOT_BUSY) ){
        prt_error_number = -2;
        prt_error_message [0] = '\0';
        strcat (prt_error_message,"Printer Error - Printer OFF
or ONLINE button not Pressed.");
        update_tau_status (2,'5');
        return;
    }

    if (status & PRT_IO_ERROR) {
        prt_error_number = -2;
        prt_error_message [0] = '\0';
        strcat (prt_error_message,"Printer Error - CHECK
PRINTER.");
        update_tau_status (2,'5');
    }
}

```


PRINTER.C

```

        return;
    }

    if (!(status & PRT_SELECTED)) {
        prt_error_number = -3;
        prt_error_message [0] = '\0';
        strcat (prt_error_message, "Printer Error - CHECK
PRINTER.");
        update_tau_status (2, '5');
        return;
    }

    if (!( (status & PRT_SELECTED) && (status & PRT_NOT_BUSY) )){
        prt_error_number = -4;
        prt_error_message [0] = '\0';
        strcat (prt_error_message, "Printer Error - CHECK
PRINTER.");
        update_tau_status (2, '5');
        return;
    }

    if ( (prt_fp = fopen ("LPT1", "wb+")) == NULL) {
        prt_error_number = -5;
        prt_error_message [0] = '\0';
        strcat (prt_error_message, "Printer Error - ERROR
WRITING TO PRINTER!");
        update_tau_status (2, '5');
        return;
    }
    prt_error_number = 0;
    prt_error_message [0] = '\0';
    strcat (prt_error_message, "Printing Agreement.");
    print_report (program_number, lost_phone);
    fclose (prt_fp);
    return;
}

print_time (t)
char t[];
{
    fprintf (prt_fp, "%c", t[0]);
    fprintf (prt_fp, "%c", t[1]);
    fprintf (prt_fp, ":");
    fprintf (prt_fp, "%c", t[2]);
    fprintf (prt_fp, "%c", t[3]);
    fprintf (prt_fp, "%c", t[4]);
}

print_phone (p)
char p[];
{
    int i;
    fprintf (prt_fp, "(");

```


PRINTER.C

```

    fprintf (prt_fp,"%c",p[0]);
    fprintf (prt_fp,"%c",p[1]);
    fprintf (prt_fp,"%c",p[2]);
    fprintf (prt_fp,"");
    fprintf (prt_fp," ");
    for (i=4;i<=11;++i)
        fprintf (prt_fp,"%c",p[i]);
}

/*-----
length ended in NULL or SPACE
-----*/
int g_length (s)
char s[];
{
    int i;
        i = 0;
        while ( (s[i] != ' ') && (s[i]) ) i++;
        return i;
}

/*-----
format_phone_number : (xxx) xxx-xxxx etc...
-----*/
char *format_phone_number (char *s) {
    char s1[80];
        /* (xxx) xxx-xxxx */
        if (strlen (s) == 10) {
            s1[0] = '(';
            s1[1] = '\0';
            strcat (s1,s,3);
            s1[4] = ')';
            s1[5] = ' ';
            s1[6] = s[3];
            s1[7] = s[4];
            s1[8] = s[5];
            s1[9] = '-';
            s1[10] = s[6];
            s1[11] = s[7];
            s1[12] = s[8];
            s1[13] = s[9];
            s1[14] = '\0';
        } else
            if ((strlen (s) == 11) && (s[10] != '*') ){
                s1[0] = '(';
                s1[1] = '\0'; /* 1-xxx-xxx-xxxx */
                s1[1] = s[1]; /* skip 1 in front of call */
                s1[2] = s[2];
                s1[3] = s[3];
                s1[4] = ')';
                s1[5] = ' ';
                s1[6] = s[4];
                s1[7] = s[5];
            }
}

```


PRINTER.C

```

s1[8] = s[6];
s1[9] = '-';
s1[10] = s[7];
s1[11] = s[8];
s1[12] = s[9];
s1[13] = s[10];
s1[14] = '\0';
} else
if ((strlen (s) == 11) && (s[10] == '*') ){
    s1[0] = '('; /* XXX-XXX-XXXX* */
    s1[1] = '\0'; /* roamer with no 1 in front */
    s1[1] = s[0]; /* no 1 in front of call */
    s1[2] = s[1];
    s1[3] = s[2];
    s1[4] = ')';
    s1[5] = ' ';
    s1[6] = s[3];
    s1[7] = s[4];
    s1[8] = s[5];
    s1[9] = '-';
    s1[10] = s[6];
    s1[11] = s[7];
    s1[12] = s[8];
    s1[13] = s[9];
    s1[14] = s[10];
    s1[15] = '\0';
} else
if ((strlen (s) == 12) && (s[11] == '*') ){
    s1[0] = '('; /* 1-XXX-XXX-XXXX* */
    s1[1] = '\0'; /* roamer with a 1 in front */
    s1[1] = s[1]; /* 1 in front of call skip it*/
    s1[2] = s[2];
    s1[3] = s[3];
    s1[4] = ')';
    s1[5] = ' ';
    s1[6] = s[4];
    s1[7] = s[5];
    s1[8] = s[6];
    s1[9] = '-';
    s1[10] = s[7];
    s1[11] = s[8];
    s1[12] = s[9];
    s1[13] = s[10];
    s1[14] = s[11];
    s1[15] = '\0';
} else
if (strlen (s) == 7) {
    strncpy (s1,s,3);
    s1[3] = '-';
    s1[4] = s[3];
    s1[5] = s[4];
    s1[6] = s[5];
    s1[7] = s[6];
    s1[8] = '\0';
}

```


PRINTER.C

```

    } else
    if ((strlen (s) == 8) && (s[7] == '*') ) {
        strncpy (s1,s,3);
        s1[3] = '-';
        s1[4] = s[3];    /* XXX-XXXX* local roamer */
        s1[5] = s[4];
        s1[6] = s[5];
        s1[7] = s[6];
        s1[8] = s[7];
        s1[9] = '\0';
    } else {
        strcpy (s1,s);
    }
    return s1;
}

```

```

/*-----
--
add_up_total_minutes
-----
*/
float add_up_total_minutes (gbaserec rec) {
    int i;
    float total;
    record_type *call;

    total = 0;
    i = 1;
    if (rec.attached_records == 0)
        return 0;
    while (i <= rec.attached_records) {
        call = g_get_call (rec,i);
        total += call->length;
        ++i;
    }
    return total;
}

```

```

print_report(int program_number, int lost_phone)
{
    float temp;
    float long_dist,access_chgs;
    int i;
    record_type *a_call_rec;
    char s[80];

    current_printer_position = 1;
    line_count = 0;

    print_newline(1);
    print_string ("\x0E\x1B\x34"); /* Bold Red */
    print_string ("      TELEMAC");
}

```


PRINTER.C

```
print_newline(1);
print_string ("          CELLULAR");
print_newline(1);
print_string ("          CORPORATION");
print_string ("\x14\x18\x35"); /* normal Black */
print_newline(2);
print_string ("          Cellular Phone Rental");
print_newline(1);
print_string ("          -----");
print_newline(1);
if (program_number == 1)
    print_string ("          ** Opening Agreement **");
if (program_number == 2)
    print_string ("          ** Ending Agreement **");
if (program_number == 0)
    print_string ("          ** Updated Agreement **");
print_newline(2);
print_string ("Agency : ");
print_section (controlrec.location_name,28);
print_newline(1);
print_string ("TAU id : ");
print_section (controlrec.tau_id,4);
print_newline (1);
print_string ("Phone Number : ");
print_section (controlrec.voice_phone_num,12);
print_newline (2);
print_string ("Payment Type : ");
    if (strcmp (agreemntrec.credittype,"AE",2)==0) print_string
("American Express");
    if (strcmp (agreemntrec.credittype,"VI",2)==0) print_string
("Visa");
    if (strcmp (agreemntrec.credittype,"MC",2)==0) print_string
("Master Card");
    if (strcmp (agreemntrec.credittype,"CB",2)==0) print_string
("Carte Blanche");
    if (strcmp (agreemntrec.credittype,"DC",2)==0) print_string
("Diners Club");
    if (strcmp (agreemntrec.credittype,"DI",2)==0) print_string
("Discover Card");
    if (strcmp (agreemntrec.credittype,"CA",2)==0) print_string
("CASH");
    if (strcmp (agreemntrec.credittype,"CK",2)==0) print_string
("CHECK");
    if (strcmp (agreemntrec.credittype,"NO",2)==0) print_string
("NONE");
    print_newline (1);
    print_string("Number : ");
    print_section(agreemntrec.creditno, 19);
    print_newline(1);
    print_string("Expires : ");
    print_string(fmt_date(agreemntrec.expiredate, "MM/DD/YY"));
    print_newline(1);
    print_string ("Approval Code : ");
    print_section (agreemntrec.approved,8);
```


PRINTER.C

```

print_newline(2);
print_string ("Agreement Number : ");
print_string ("\x1B\x45"); /* Emphasized */
print_section(agreementrec.agreeno, 13);
print_string ("\x1B\x46");
print_newline(1);
print_string ("Phone Number : ");
print_string ("\x1B\x34"); /* red */
print_phone(agreementrec.curphoneno);
print_string ("\x1B\x35");
print_newline(1);
print_string ("Rental Date : ");
print_string(fmt_date(agreementrec.rentaldate, "MM/DD/YY"));
print_newline(1);
print_string ("Rental Time : ");
print_string (agreementrec.timeout);
print_newline(1);
if (program_number == 2) {
    print_string ("Return Date : ");
    print_string(fmt_date(agreementrec.actrtndate, "MM/DD/YY"));
    print_newline(1);
    print_string ("Returned Time: ");
    print_string (agreementrec.timein);
}
print_newline(1);
print_string ("Rented By : ");
print_string(agreementrec.origperson);
if (program_number == 2) {
    print_string ("Returned To : ");
    print_string (agreementrec.preparedby);
}
print_newline(2);
print_string ("Customer :");
print_newline(1);
print_section(agreementrec.custname, 25);
/* if (agreementrec.company[0] != ' ') {
    print_newline(1);
    print_section(agreementrec.company, 25);
}
*/
print_newline(1);
print_section(agreementrec.custaddr1, 25);
/*
if (agreementrec.custaddr2[0] != ' ') {
    print_newline(1);
    print_section(agreementrec.custaddr2, 25);
}
*/
print_newline(1);
zap_trailing_spaces (agreementrec.custcity);
print_section(agreementrec.custcity, strlen (agreementrec.custcity));
print_string (" ");
print_section(agreementrec.custstate, 2);
print_string (" ");

```


PRINTER.C

```

print_section(agreementrec.custzipcd, 10);
/*
if (agreementrec.busphone[0] != ' ') {
    print_newline(1);
    print_string ("Business Phone : ");
    print_section(agreementrec.busphone, g_length(agreementrec.busphon
e));
}
*/
print_newline(1);
print_string ("Home Phone : ");
print_section(agreementrec.homephone, 12);
print_newline(1);
print_string ("Drivers License : ");
print_section(agreementrec.licenseno, 10);
print_newline(3);
print_string ("=====");
print_newline (1);
print_string ("\x1B\x45"); /* Emphasized Characters */
print_string ("EQUIPMENT RENTED");
print_string ("\x1B\x46"); /* Emphasized Characters */
print_newline(1);
print_string ("-----");
print_newline(1);
print_string ("Phones Rented : 1");
print_tab(27);
if (program_number == 2) {
    if (lost_phone) {
        print_string ("Returned : 0");
    } else print_string("Returned : 1");
} else print_string("Returned : 0");
print_newline(1);
print_string ("Batteries Rented: ");
print_string(fmt_dbl(agreementrec.nobatrent, "Z9"));
print_tab (27);
print_string("Returned : ");
print_string(fmt_dbl(agreementrec.nobatrttn, "Z9"));
/*
print_newline(1);
print_string ("Cases Rented : ");
print_string(fmt_dbl(agreementrec.nocasrent, "Z9"));
print_tab (27);
print_string ("Returned : ");
print_string(fmt_dbl(agreementrec.nocasrttn, "Z9"));
*/
print_newline(1);
print_string ("Chargers Rented : ");
print_string(fmt_dbl(agreementrec.nochgrent, "Z9"));
print_tab (27);
print_string ("Returned : ");
print_string(fmt_dbl(agreementrec.nochgrtn, "Z9"));
print_newline (2);
print_string ("Phone Charge/Minute :");
print_string(fmt_dbl(controlrec.charge_per_minute, "Z,ZZ9.99-"));

```


PRINTER.C

```

103
print_newline (1);
print_string (" + Long Distance");
print_newline (1);
print_string ("Roaming Charges/Minute :");
print_string (fmt_dbl(controlrec.roam_chg_per_min, "Z,ZZ9.99-"));
print_newline (1);
print_string ("Roaming Charges/Day :");
print_string (fmt_dbl(controlrec.roam_chg_per_day, "Z,ZZ9.99-"));
print_newline (1);
print_string ("Phone Charge/Day :");
temp = controlrec.phone_daily_chg;
if (program_number == 2)
    print_string (fmt_dbl
(agreemntrec.phochgday, "Z,ZZ9.99-"));
if ( (program_number == 1) || (program_number == 0) ) {
if (agreemntrec.discount > 0) {
temp = (temp - (temp * agreemntrec.discount/100));
print_string(fmt_dbl(temp, "Z,ZZ9.99-"));
print_string ("\x1B\x34"); /* red */
print_string (" *DISCOUNT");
print_string ("\x1B\x35");
} else
print_string(fmt_dbl(temp, "Z,ZZ9.99-"));

/* NOTE the variable costfax has been overridden to */
/* to represent LDW charges per day */
print_newline (1);
print_string ("LDW Charge/Day :");
print_string (fmt_dbl (controlrec.ldw_daily_chg,
"Z,ZZ9.99-"));
print_newline (1);
if (agreemntrec.remarks5[0] == 'Y') {
print_string ("LDW Accepted : [YES]");
} else print_string ("LDW Accepted : [NO]");
print_newline (2);
print_string ("Initial Meter Reading:");
print_newline (1);
print_string ("Hours :");
print_string(fmt_dbl(agreemntrec.hoursout, "999"));
print_newline (1);
print_string ("Minutes :");
print_string(fmt_dbl(agreemntrec.minutesout, "99"));
print_newline (2);
}
print_newline (1);
print_string ("=====");
print_newline (1);

if (program_number == 2) {
print_newline (2);
print_string ("-----");
print_newline (1);
print_string ("\x1B\x45"); /* Emphasized Characters */
print_string ("Call Record:");

```


PRINTER.C

```

print_string ("\x1B\x46"); /* Emphasized Characters */
print_newline (1);
print_string ("Date:          Time:          Number:");
print_newline (1);
print_string ("Length:          L/D Chg:          Access Chg:");
print_newline (1);
print_string ("-----");
print_newline (1);
long_dist = 0;
access_chgs = 0;

for (i=1;i<=call_rec.attached_records;i++) {
    a_call_rec = g_get_call (call_rec,i);
    print_string(fmt_date(a_call_rec->date,
"MM/DD/YY"));

    print_string(" ");
    print_string(a_call_rec->start_time);
    print_string(" ");
    print_string
(format_phone_number(a_call_rec->number));
    print_newline (1);
    print_string (fmt_dbl (a_call_rec->length, "Z99"));
    if (strcmp (a_call_rec->number,"ROAMING",7) == 0) {
        print_string (" Days");
    } else print_string (" Mins");
    print_string (" ");
    print_string(fmt_dbl (a_call_rec->long_dist_cost,
"Z,ZZ9.99-"));

    print_string (" ");
    print_string(fmt_dbl (a_call_rec->base_cost,
"Z,ZZ9.99-"));

    long_dist = long_dist + a_call_rec->long_dist_cost;
    access_chgs = access_chgs + a_call_rec->base_cost;
    print_newline (1);
}
print_string ("-----");
print_newline (1);
print_string ("TOTAL ACCESS CHARGES          : ");
print_string (fmt_dbl (access_chgs, "Z,ZZ9.99-"));
print_newline (1);
print_string ("TOTAL LONG DISTANCE CHARGES: ");
print_string (fmt_dbl (long_dist,"Z,ZZ9.99-"));
print_newline (1);
print_string ("-----");
print_newline (2);

print_string ("Days Used : ");
print_string(fmt_dbl(agreementrec.daysused, "99"));
print_newline(1);
print_string ("Minutes used : ");
print_string(fmt_dbl(add_up_total_minutes (call_rec),"99999"));
print_newline(1);
print_string ("Days Usage Charge : ");
print_tab (30);

```


PRINTER.C

```

print_string(fmt_dbl(agreementrec.dlyphochg, "Z,ZZ9.99-"));
if (agreementrec.discount > 0){
    print_newline (1);
    print_tab (14);
    print_string ("\x1B\x34"); /* red */
    print_string ("*DISCOUNTED DAILY RENTAL");
    print_string ("\x1B\x35"); /* red */
}
if (agreementrec.discount != 0.0) {
    print_newline(1);
    print_string ("Discount : ");
    print_tab (33);
    if (agreementrec.discount > 0) {
        print_string ("%");
        print_string
(fmt_dbl(agreementrec.discount,"Z99"));
    } else print_string ("% 0");
}
print_newline(1);
print_string ("Minutes Usage Charge : ");
print_tab (30);
print_string(fmt_dbl(agreementrec.minphochg, "Z,ZZ9.99-"));
/*
    print_newline(1);
    print_string ("Damage Charge :");
    print_tab (30);
    print_string(fmt_dbl(agreementrec.damagechg, "Z,ZZ9.99-"));
*/

print_newline(1);
print_string ("Unreturned Equipment Charge: ");
print_tab (30);
print_string(fmt_dbl(agreementrec.equipchg, "Z,ZZ9.99-"));
if (agreementrec.adjustment != 0.0) {
    print_newline (1);
    print_string ("Adjustment : ");
    print_tab (30);
    print_string("<");
    print_string(fmt_dbl(agreementrec.adjustment,
"Z,ZZ9.99-"));
    print_string(">");
}
print_newline (1);
print_string ("LDW Chgs : ");
print_tab (30);
print_string (fmt_dbl(agreementrec.ldw_charges, "Z,ZZ9.99-") );
print_newline(1);
print_string ("Subtotal : ");
print_tab (30);
print_string(fmt_dbl(agreementrec.subtotal, "Z,ZZ9.99-"));
print_newline(1);
print_string ("-----");
print_newline(1);
print_string ("Total Tax: ");
print_tab (30);
print_string(fmt_dbl(agreementrec.total_tax, "Z,ZZ9.99-"));

```


PRINTER.C

```

print_newline(1);
print_string ("-----");
print_newline(1);
    print_string ("\x1B\x45"); /* Emphasized Characters */
    print_string ("TOTAL BILL :");
    print_string ("\x1B\x46"); /* Emphasized Characters */
    print_tab (34);
    print_string(fmt_dbl(agreementrec.netdue, "Z,ZZ9.99-"));
    print_newline(1);
    print_string ("Amount Paid :");
    print_tab (30);
    print_string (fmt_dbl(agreementrec.amtpaid, "Z,ZZ9.99-"));
    print_newline(1);
    print_string ("=====");
    print_newline (1);
    print_string ("\x1B\x45");
    print_string ("BALANCE DUE :");
    print_tab (30);
    print_string (fmt_dbl(agreementrec.amtowed, "Z,ZZ9.99-"));
    print_string ("\x1B\x46");
    print_newline (1);
    print_string ("=====");
    print_newline (3);
    print_string("Remarks : ");
    print_newline(1);
    print_section(agreementrec.remarks1, 25);
    print_newline (1);
    print_section(agreementrec.remarks2, 25);
    print_newline (1);
    print_section(agreementrec.remarks3, 25);
}

    print_string ("\x1B\x34"); /* red */
    print_newline (1);
    print_string ("Customer has read and agreed to the");
    print_newline (1);
    print_string ("terms and conditions as stated above");
    print_newline (1);
    print_string ("and on the reverse side of this receipt.");
    print_string ("\x1B\x35"); /* red off */
    print_newline (2);
    print_string ("-----");
    print_newline (1);
    print_string ("Signature:");
    print_newline (8);
}

/*
 * Output blank lines until next page
 */
prt_eject()
{
    fprintf (prt_fp, "\014");
}

```


PRINTER.C

RTBFUNC.C

/*-----
 -
 MODULE rtbfunc.c

PURPOSE: This module does the initialization of the phone for the real-time billing event. Upon a return of a phone, the realtime.c MODULE performs the necessary actions to return and calculate phone charges. Together these two modules are the realtime billing system.

Written By : Greg McGregor 1990

REVISED: What was revised?
 GMM 7-30-1991 Nothing

 */

#include <stdio.h>
 #include <stdlib.h>
 #include <gkeys.h>
 #include <bios.h>
 #include <time.h>
 #include <windows.h>
 #include <gbase.h>
 #include <bench.h>
 #include <proc.io>
 #include <agrio.h>
 #include <agreev3.h>
 #include <extnvar.h>
 #include <extscrns.h>
 #include <rtb.h> /* realtime billing definitions */

#define TIMED_OUT_X 1000 /* = 1.0+ sec time out */

/*-----
 move_hl: move high nibble to low nibble and set high to 0x00
 -----*/

char move_hl (char nib)
 {
 nib = nib >> 4; /* move upper 4 bits to lower 4 bits */
 nib = nib & 0x0F; /* set high nibble to 0x00 */
 return nib;
 }

/*-----
 move_lh: move low nibble to high nibble;
 -----*/

char move_lh (char nib)
 {
 nib = nib << 4;

RTBFUNC.C

```
nib = nib & 0xF0; /*set low nib to 0 */
return nib;
}

/*-----
set_rtb_port
-----*/
set_rtb_port (int port)
{
    RTB_PORT = port;
}

/*-----
rtb_null_field: put nulls in every byte in field
-----*/
rtb_null_field (char *f,int len)
{
    int i;
    for (i=0;i<len;i++) {
        f[i] = '\0';
    }
}

/*-----
to_digit: converts a number 0-9 to a char 0-9
-----*/
to_digit (char num)
{
    char zero = '0';
    char n1;

    n1 = num;
    if (num == 0x0A) return '0';    /* numbers in BCD format */
    num = num + zero;
    if (n1 == 0x0B) num = '*';
    if (n1 == 0x0C) num = '#';
    return num;
}

/*-----
open_rtb_port
-----*/
int open_rtb_port()    /* INCOMPLETE */
{
    bioscom (0, SETTINGS, RTB_PORT);
    return (1);
}

/*-----
rtb_error: display rtb error
-----
*/
```


RTBFUNC.C

```

rtb_error (int e)
{
    wintype win;
    char msg[80];

    switch (e) {
        case 0 : strcpy (msg, "ERROR 0 : General Failure!");
                    break;
        case -1 : strcpy (msg, "ERROR 1 : Database Files Not
Found!");
                    break;
        case -2 : strcpy (msg, "ERROR 2 : Couldn't Open RTB Port!");
                    break;
        case -3 : strcpy (msg, "ERROR 3 : Data Download Error!");
                    break;
        case -4 : strcpy (msg, "ERROR 4 : Data Parsing Error!");
                    break;
        case -5 : strcpy (msg, "ERROR 5 : Communication Error!");
                    break;
        case -6 : strcpy (msg, "ERROR 6 : State Transition Error!");
                    break;
        case -7 : strcpy (msg, "ERROR 7 : No Phone In CTI!");
                    break;
        case -8 : strcpy (msg, "ERROR 8 : Can't Unlock Phone!");
                    break;
        case -9 : strcpy (msg, "ERROR 9 : Can't Get Cellular Phones
Phone Number!");
                    break;
        case -10 : strcpy (msg, "ERROR 10: Can't Get Clock
Information From Phone!");
                    break;
        case -11 : strcpy (msg, "ERROR 11: Can't Retrieve Call
Counter!");
                    break;
        case -12 : strcpy (msg, "ERROR 12: Can't End CTI
Transmission!");
                    break;
        case -13 : strcpy (msg, "ERROR 13: Can't Reset Phone's
Memory Pointer!");
                    break;
        case -14 : strcpy (msg, "ERROR 14: Can't Reset Phone Meter!");
                    break;
        case -15 : strcpy (msg, "ERROR 15: Can't Set/Get Phone's
Clock Chip!");
                    break;
        case -16 : strcpy (msg, "ERROR 16: Can't Get Number Of
Calls!");
                    break;
        case -17 : strcpy (msg, "ERROR 17: Can't Reset Call
Counter!");
                    break;
        case -18 : strcpy (msg, "ERROR 18: Can't Lock Phone!");
                    break;
        case -19 : strcpy (msg, "ERROR 19: Couldn't Power Down

```


RTRFUNC.C

```

Phone!");
        break;
    case -20: strcpy (msg,"ERROR 20: Can't Get Phone's Memory
Pointer!");
        break;
    case -21: strcpy (msg,"ERROR 21: Phone Not Rented Out!");
        break;
    case -22: strcpy (msg,"ERROR 22: General Failure!");
        break;
    case -23: strcpy (msg,"ERROR 23: This Phone Has Not Been
Logged In At This Site!");
        break;
    case -24: strcpy (msg,"ERROR 24: Could NOT get meter
reading from phone!");
        break;
    case -25: strcpy (msg,"ERROR 25: Damaged Phone!");
        break;
    case -26: strcpy (msg,"ERROR 26: Operator Aborted!");
        break;
    case -27: strcpy (msg,"ERROR 27: Command Failed!");
        break;
    case -28: strcpy (msg,"ERROR 28: Transfer Timeout!");
        break;
    case -29: strcpy (msg,"ERROR 29: Lost Phone!");
        break;
    }
    strcpy (errmsg,msg);
    errrtn(errmsg);
}

/*-----
wait_command
-----
*/
wait_command ()
{
    delay (1000); /* wait 1000 milliseconds between commands */
}

/*-----
wait_byte :wait time to send a byte
-----
*/
wait_byte ()
{
    delay (75); /* wait 50 milliseconds between data bytes send */
}

/*-----
wait_error : wait time for error
-----

```


RTBFUNC.C

```

-*/
wait_error () {
    delay (1000);
    flush_port ();
}

/*-----
wait_receive : wait time to receive byte, delay
-----
-*/
wait_receive ()
{
    delay (1); /* delay 1 milliseconds */
}

/*-----
flush_port : wait and retrieve all data coming, time out after 1 sec
             clears any data hanging around the port
-----
-*/
flush_port ()
{
    int i,t,stat,in;
    t = 0;
    while (t<TIMED_OUT_X) { /* wait ~ 1 sec and time_out */
        stat = bioscom (3,0,RTB_PORT);
        if (stat & DATA_READY) {
            in = bioscom (2, 0, RTB_PORT);
            t = 0;
        } else {
            wait_receive ();
            ++t;
        }
    } /* timed out no data left coming */
}

/*-----
shift_left : shift s left 1 length 1 put null in 1
-----*/
shift_left (char *s,int l)
{
    int i;
    for (i=1;i<l;i++)
        s[i-1] = s[i];
    s[l-1] = '\0';
}

```


RTBFUNC.C

```

/*-----
convert_to_phone_time
ARGS:  struct tm a_time
-----*/
convert_to_phone_time (unsigned char converted[])
{
int i;
char data[10];
unsigned char ch,ch2;
time_t timer;
struct tm *tblock;

    timer = time (NULL);
    tblock = localtime (&timer);

    data[0] = 0; /* set .1 and .01 sec to .01 */
                /* convert seconds to BCD format */
    ch = (char ) (tblock->tm_sec / 10); /* 10 secs */
    ch = move_lh (ch); /* shift low nib to high nib */
    ch = ch | ( (tblock->tm_sec) - ((tblock->tm_sec / 10)* 10 ) );
    data[1] = ch;
                /* convert minutes */
    ch = (char ) (tblock->tm_min / 10); /* get 10's place in mins */
    ch = move_lh (ch); /* truncates 1's place */
    ch = ch | ( (tblock->tm_min) - ((tblock->tm_min / 10) * 10) );
    data[2] = ch;
                /* convert hours, set as 12 hour mode */
    if (tblock->tm_hour < 12) { /* 0 to 11am */
        ch = 0; /* set all bits to 0 */
        ch = (char ) tblock->tm_hour;
        if (ch == 0) {
            ch = 12; /* for 12 AM */
            tblock->tm_hour = 12;
        }
        data[3] = ch;
    } else
    if (tblock->tm_hour < 24) {
        ch = 0;
        ch = (char ) tblock->tm_hour;
        if (ch != 12) {
            ch = ch - 12; /* convert back to 12 hour */
        } else {
            ch = 12;
        }
        ch = ch | BIT6; /* set pm bit on */
        data[3] = ch;
    }

    /* turn on 12 hour mode */
    data[3] = data[3] | BIT8; /* bled 1 - 8 */

    /* convert day of week */
    ch = (char ) (tblock->tm_wday + 1); /* sunday = 0 so bump by 1 */
    data[4] = ch;
    data[4] = data[4] | BIT5; /* Turn on reset bit */

```


RTBFUNC.C

```

data[4] = data[4] & 0xDF; /* Turn off bit 6 oscillator bit */

/* convert day of month */
ch = (char ) (tblock->tm_mday / 10);
ch = move_lh (ch); /* move tens to upper nibble */
ch = ch | ( (tblock->tm_mday) - ((tblock->tm_mday / 10) * 10) );
data[5] = ch;

/* convert month */
ch = (char ) (tblock->tm_mon + 1) ; /* month starts at 0 so bump */
ch = ch / 10;
ch = move_lh (ch);
ch = ch | ( (tblock->tm_mon + 1) - (((tblock->tm_mon + 1) / 10) *
10) );
data[6] = ch;

/* convert year */
ch = (char ) (tblock->tm_year / 10);
ch = move_lh (ch);
ch = ch | ( (tblock->tm_year) - ((tblock->tm_year / 10) * 10) );
data[7] = ch;

moveX (converted,data,8);
}

```


SERVER.C

```
/*-----
-----
```

server For GVN Network

PURPOSE:

Waits for a host to log on and preforms functions.

Written By: Greg McGregor 1990

REVISED: What was revised?

GMM 7-30-1991	Nothing
GMM 8-13-1991	Started the delete execute commands on TAU
GMM 8-14-1991	Finished Delete and Execute commands on TAU V1.50
GMM 8-26-1991	Adjusted version numbers to 1.52
GMM 9-9-1991	Won't hang on initializing modem

```
-----
--*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <process.h>
#include <time.h>
#include <window.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <mem.h>
#include <font1.h>
#include <sys\stat.h>
#include <io.h>
#include "asiports.h"
#include "xfer.h"
#include "ibmkeys.h"
#include "gf.h"
```

```
#include <windows.h>
#include <misc.h>
#include <time.h>
#include <gbase.h>
#include <extnvar.h>
```

```
void status_routine (char *m); /* local commands */
void transfer_status (XFER *b);
char calc_CRC (char *s,int len);
int get_xchar ();
int set_answer ();
struct tm far *get_life ();
void start_server ();
void end_server ();
void run_server ();
void hang_up ();
```


SERVER.C

```

void hang_up1 ();
int is_ring ();
int init_modem ();

/*
 * Window Defs
 */
windef comm_win =
{10,12,70,17,White,Blue,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
  White,Blue};
windef wait_win = {10,4,70,6,White,Red,FALSE,FALSE,FALSE,TRUE,SINGLEFR
AME,
  White,Red};
windef status_win =
{5,20,75,22,White,Blue,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
  White,Blue};

/*
 * Window Types
 */
wintype comm_wt,wait_wt,status_wt;

#define FULL 1
#define HALF 0
#define MODE ASINOUT|BINARY|NORMALRX
#define RXLEN 1024
#define TXLEN 1024
#define SECONDS 5
#define TRUE 1
#define FALSE 0
#define ECHO 0
#define SPEAKER OFF

char ACK_CHAR = 0x20;
char NAK_CHAR = 0x21;
char LOG_OUT = 0x22;
char SEND_COMMAND = 0x23;

int PORT;
int BAUD = 2400;          /* Hotels are all at 2400 Baud */
int PARITY = P_NONE;     /* No Parity */
char PHONE_NUMBER [20];  /* phone number to call */
int STOP_BITS = 1;
int WORD_LENGTH = 8;
int DUPLEX = FULL;
char command_list[80];
char file_name[80];

```

```

/*-----

```


SERVER.C

```

---
main:
-----
---*/
/**
main (int argc, char *argv[])
{
    int done;
        done = FALSE;
        init_windows ();
        check_args (argc,argv);
        comm_wt = windowopen (&comm_win);
        setttitle (comm_wt,"GVN Server V1.55",CenterUpperTitle);
        PORT = atoi (argv[1]) - 1;
    while (!done) {
        asiclear (PORT,ASINOUT);
        open_port ();
        init_modem ();
        if (set_answer()) {
            wait_for_commands ();
        } else done = TRUE;
        hang_up ();
    }
}
*****/
/*-----
set_gvn_port
-----*/
set_gvn_port (int port)
{
    PORT = port;
}

/*-----
start_server
-----*/
void start_server ()
{
    comm_wt = windowopen (&comm_win);
    setttitle (comm_wt,"GVN Server V1.55",CenterUpperTitle);
    clrscr ();
    gotoxy (15,2);
    cprintf ("GVN Loading -> ");
    textbackground (Black);
    gotoxy (30,2);
    cprintf (" ");
    gotoxy (30,2);
    open_port ();
    cprintf ("%c%c",219,219);
    init_modem ();
    cprintf ("%c%c",219,219);
    HMSetWaitTimeForCarrier (PORT,30);
    cprintf ("%c%c",219,219);
    HMSetAutoAnswerRingCount (PORT,1);
}

```


SERVER.C

```

cprintf ("%c%c",219,219);
windowclose (comm_wt);
}

/*-----*/
end_server
/*-----*/

void end_server ()
{
    comm_wt = windowopen (&comm_win);
    setttitle (comm_wt,"WAIT!",CenterUpperTitle);
    hang_up ();
    windowclose (comm_wt);
}

/*-----*/
end_server1
/*-----*/

void end_server1 ()
{
    comm_wt = windowopen (&comm_win);
    setttitle (comm_wt,"WAIT!",CenterUpperTitle);
    hang_up1 ();
    windowclose (comm_wt);
}
*/

/*-----*/
is_ring
/*-----*/

int is_ring ()
{
    return (iscd (PORT,IMMEDIATE));
}

/*-----*/
far_run_server
/*-----*/

void run_server ()
{
    wait_wt = windowopen (&wait_win);
    setttitle (wait_wt,"GVN Network Active",CenterUpperTitle);
    clrscr ();
    textcolor (White+Blink);
    cprintf ("                Please Wait Until Host Is Finished!");
    textcolor (White);
    comm_wt = windowopen (&comm_win);
    setttitle (comm_wt,"GVN Server V1.55",CenterUpperTitle);
    /*
    HMSetCarrier (PORT,ON);
    HMAnswer (PORT);
    HMSetHookSwitch (PORT,OFFHOOK);
    if (!wait_for_commands ()) {

```


SERVER.C

```

use (wait_wt);
windowclose (wait_wt);
use (comm_wt);
windowclose (comm_wt);
}
}

/*-----
set_answer
-----*/

/
int set_answer ()
{
int count;
char ch;

use (comm_wt);
clrscr ();
cprintf("  - *  Waiting For TELEMAT Host Connect!");
HMSetEscapeCode (PORT,ESC);
HMSetWaitTimeForCarrier (PORT,30);
while (HMGetIncomingRingCount (PORT) < 1) {
    if (kbhit ()) {
        ch = getch ();
        if (ch == 0x1B) /* an ESC key */
            return FALSE;
    }
}
HMSetCarrier (PORT,ON);
HMAAnswer (PORT);
HMSetHookSwitch (PORT,OFFHOOK);
return TRUE;
}

/*-----
connected : PREDICATE is connected to site
-----*/

int connected ()
{
return iscd (PORT,CUMULATIVE);
}

/*-----
recieve_command
-----*/

char recieve_command (char c)
{
int stat;
int trys = 0;

while ( ( (stat = get_xchar ()) != c) && (trys < 100) ) {
    ++ trys;
    send_xchar (NAK_CHAR);
}
}

```


SERVER.C

```

    }
    if (stat == c ) {
        send_xchar (ACK_CHAR);
        return TRUE;
    }
    return FALSE; /* error */
}

/*-----
send_command-----*/
char send_command (char c)
{
    int stat;
    int trys;

    trys = 0;
    send_xchar (c);
    do {
        stat = get_xchar ();
        if (stat != 0) cprintf ("%g",stat);
        ++trys;
        send_xchar (c);
    } while ( (stat != ACK_CHAR) && (trys < 30) );

    if (stat == ACK_CHAR)
        return TRUE;
    return FALSE;
}

/*-----
wait_for_commands-----*/
int wait_for_commands ()
{
    int stat,a_command,done,B_connected,trys;
    int idle_time = 0;
    int fd,HOST_LOCKED_MODE = FALSE;
    char s[20];
    B_connected = FALSE;
    done = FALSE;
    trys = 0;
    while (!connected ()) ;
    asiclear (PORT,ASINOUT);
    while (!done) {
        use (comm_wt);
        clrscr ();
        cprintf ("-* Sending Job Request [ ]");
        trys = 0;
        while (!send_command (SEND_COMMAND) ) {
            clrscr ();
            ++trys;

```


SERVER.C

```

cprintf ("-* Sending Job Request [%d]", trys);
if (trys >= 3) return;
}
clrscr ();
cprintf ("-* Waiting For A Command");
stat = get_xchar () ; /* wait for a command */
if (stat == 0) ++idle_time;
if (stat == 0x02) { /* request for file transfer*/
    clrscr ();
    cprintf ("-* Host Requesting An UPLINK!");
    send_xchar (ACK_CHAR);
    file_send();
    idle_time = 0;
} else
if (stat == 0x01) {
    clrscr ();
    cprintf ("-* Host DOWNLINKING A File");
    timer (TICKS_PER_SECOND);
    send_xchar (ACK_CHAR);
    file_receive();
    idle_time = 0;
} else
if (stat == 0x03) {
    clrscr ();
    cprintf ("-* Host Requested An Archive!");
    send_xchar (ACK_CHAR);
    archive_database ();
    idle_time = 0;
} else
if (stat == 0x05) {
    clrscr ();
    cprintf ("-* Date/Time Verification...");
    send_xchar (ACK_CHAR);
    date_time_set ();
    idle_time = 0;
} else
if (stat == 0x08) {
    clrscr ();
    cprintf ("-* Sending Agreement Records To Host");
    send_xchar (ACK_CHAR);
    send_agreement();
    idle_time = 0;
} else
if (stat == 0x09) {
    clrscr ();
    cprintf ("-* Sending Phone File To Host");
    send_xchar (ACK_CHAR);
    send_phone ();
    idle_time = 0;
} else
if (stat == 0x0A) {
    clrscr ();
    cprintf ("-* Host Requesting A Reboot!");
    send_xchar (ACK_CHAR);

```


SERVER.C

```
        reboot ();
    } else
    if (stat == 0x08) {
        clrscr ();
        cprintf ("-* Host Locking Software!");
        send_xchar (ACK_CHAR);
        end_server ();
        lock_software ();
        idle_time = 0;
        done = TRUE;
        HOST_LOCKED_MODE = TRUE;
    } else
    if (stat == 0x0C) {
        clrscr ();
        cprintf ("-* Host Requesting Serial Number!");
        send_xchar (ACK_CHAR);
        send_serial_number ();
        idle_time = 0;
    } else
    if (stat == 0x0D) {
        clrscr ();
        cprintf ("-* Receiving Monthly Vitamins!");
        send_xchar (ACK_CHAR);
        put_life ();
        idle_time = 0;
    } else
    if (stat == 0x0E) {
        clrscr ();
        cprintf ("-* Host Unlocking Software!");
        send_xchar (ACK_CHAR);
        SYSTEM_LOCKED = FALSE;
        fd = open ("c:\\\\~\\\\", O_WRONLY|O_BINARY|O_TRUNC, S_IWRITE);
        s[0] = '1';          /* reset file flag as unlocked */
        write (fd, s, 1);
        close (fd);
        idle_time = 0;
        done = TRUE;
    } else
    if (stat == 0x0F) {
        clrscr ();
        cprintf ("-* Host Requesting A File ZAP!");
        send_xchar (ACK_CHAR);
        zap_file ();
        idle_time = 0;
    } else
    if (stat == 0x10) {
        clrscr ();
        cprintf ("-* Host Requesting A File EXECUTE!");
        send_xchar (ACK_CHAR);
        execute_file ();
        idle_time = 0;
    } else
    if (stat == 0x11) {
        clrscr ();
```


SERVER.C

```

        cprintf ("--* Host Requesting A Data LOCK!");
        send_xchar (ACK_CHAR);
        ME_LOCK = TRUE;
    } else
    if (stat == 0x12) {
        clrscr ();
        cprintf ("--* Host Requesting A Data UNLOCK!");
        send_xchar (ACK_CHAR);
        ME_LOCK = FALSE;
    } else
    if (stat == LOG_OUT){
        gotoxy (1,2);
        cprintf ("Log out!");
        done = TRUE;
    } else
        send_xchar (NAK_CHAR);

    if (idle_time > 20) {
        cprintf ("Timed out!");
        return; /* no command for 21 loops */
    }
}
return HOST_LOCKED_MODE;
}

/*-----
get_xchar: get char from line
-----*/
int get_xchar ()
{
    int stat;
    int trys = 0;

    while ( ( (stat = asigetc (PORT)) < ASSUCCESS) && (trys < 10000) ){
        ++trys;
    }
    if (trys < 10000) return stat;
    return 0;
}

/*-----
send_xchar : send a char down line
-----*/
send_xchar (char c)
{
    int stat;
    while ( (stat = asiputc (PORT,c)) < ASSUCCESS) ;
}

/*-----
get_data
-----*/

```


SERVER.C

```

int get_data (char *data)
{
    char bytes,crc,crc1;
    int i,j,k,stat,return_value;
    char temp[256],temp1[256];

    k = 3;
    while (k) {
        while ( (bytes = get_xchar ()) == 0);
        use (comm_wt);
        clrscr ();
        cprintf ("-* Bytes Coming %d ",bytes);
        i = 0;
        while (i<bytes-1) {
            temp[i] = get_xchar ();
            ++i;
        }

        crc = temp[i-1];
        temp[i-1] = '\0';
        crc1 = calc_CRC (temp,(bytes - 2));
        if (crc != crc1) {
            return_value = FALSE;
            --k;
            asiputc (PORT,NAK_CHAR);
            cprintf ("-* Retrying Data...");
        }
        if (crc == crc1 ) {
            k =0;
            return_value = TRUE;
            asiputc (PORT,ACK_CHAR);
        }
    }
    strncpy (data,temp,bytes-2);
    data[bytes-2] = '\0';
    return return_value;
}

```

```

/*-----*/
calc_CRC
/*-----*/

```

```

char calc_CRC (char *s,int len)
{
    int i,j;
    char crc;

    crc = 0;
    i = len;
    crc = s[0];
    for (j=1;j<i;j++)
        crc = crc ^ s[j];
    return crc;
}

```


SERVER.C

```

/*-----
file_receive
-----*/
file_receive ()
{
    int stat;
        stat = YmodemReceive (PORT,status_routine,NULL,ESC);
        gotoxy (1,4);
        cprintf ("File Transfer Status %d",stat);
}

/*-----
send_agreemnt
-----*/
send_agreemnt ()
{
    int stat;
        stat = YmodemSend (PORT,"agreemnt.",status_routine,NULL,ESC);
}

/*-----
send_phone
-----*/
send_phone ()
{
    int stat;
        stat = YmodemSend (PORT,"phone.",status_routine,NULL,ESC);
}

/*-----
file_send
-----*/
file_send ()
{
    int stat;
    char file_name [80];

        cprintf ("--* Data Coming...");
        if (!get_data (file_name)) {
            clrscr ();
            cprintf ("--* Couldn't Get File Name From Host");
        } else {
            clrscr ();
            cprintf ("Host Requested File '%s'",file_name);
            timer (TICKS_PER_SECOND * 3); /* wait 3 seconds before
send starts */
            stat = YmodemSend (PORT,file_name,status_routine,NULL,ESC);
        }
}

void status_routine (char *m)
{

```


SERVER.C

```
        gotoxy (1,3);
        cprintf ("                ");
        gotoxy (1,3);
        cprintf ("%s\n",m);
    }

/*-----
reboot ()
-----*/
reboot () {
    system ("reboot");
}

/*-----
send_serial_number
-----*/
send_serial_number () {
    int fd;
    int stat;
    fd = open
("serial.dat",O_WRONLY|O_TEXT|O_TRUNC|O_CREAT,S_IRREAD|S_IWRITE);
    write (fd,"Not implemented!",strlen ("Not implemented!"));
    close (fd);
    stat = YmodemSend (PORT,"serial.dat",status_routine,NULL,ESC);
}

/*-----
lock_software
-----*/
lock_software () {
    end_server ();
    lock_system ();
}

/*-----
file_exists
-----*/
int file_exists (char *s)
{
    FILE *f;
    f = fopen (s,"r");
    if (f == NULL)
        return FALSE;
    fclose (f);
    return TRUE;
}

/*-----
put_life: re-birth software
-----*/
```


SERVER.C

```

put_life ()
{
    struct tm *t;
    time_t tm;
    int fd;

    tm = time(NULL);
    t = localtime (&tm);
    t->tm_yday += 30; /* add one month to life span */
    if (t->tm_yday > 365) { /* allow for year change */
        t->tm_year++;
        t->tm_yday = t->tm_yday - 365;
    }
    fd = open ("lspan.dat", O_BINARY|O_WRONLY|O_CREAT|O_TRUNC, S_IWRITE);
    if (fd != -1) {
        write (fd, t, sizeof (struct tm));
    }
    close (fd);
}

/*-----
get_life : return Birthday of software
-----*/

struct tm life; /* global */

struct tm *get_life ()
{
    int fd;

    if (!file_exists ("lspan.dat")) {
        return NULL;
    }
    fd = open ("lspan.dat", O_BINARY|O_RDONLY, S_IREAD);
    if (fd == -1) {
        return NULL;
    }
    read (fd, &life, sizeof (struct tm));
    close (fd);
    return &life;
}

/*-----
zap_file
-----*/

zap_file () {
    int stat;
    char file_name [80];
    char command_string[255];

    gotoxy (1, 2);
    cprintf ("-* Data Coming...");

```


SERVER.C

```

    if (!get_data (file_name)) {
        clrscr ();
        cprintf ("-* Couldn't Get File Name From Host");
    } else {
        strcpy (command_string,"del ");
        strcat (command_string,file_name);
        system (command_string);
    }
}

/*-----
execute_file
-----*/
execute_file () {
    int stat;
    char file_name [80];

    gotoxy (1,2);
    cprintf ("-* Data Coming...");
    if (!get_data (file_name)) {
        clrscr ();
        cprintf ("-* Couldn't Get File Name From Host");
    } else {
        system (file_name);
    }
}

/*-----
date_time_set () : set date and time according to host
-----*/
/
date_time_set ()
{
    struct time t;
    struct date d;
    int stat1,stat2;
    char data[80];

    use (comm_wt);
    clrscr ();
    cprintf ("-* Getting Time...");
    stat1 = get_data (data);
    if (stat1) memcpy (&t,data,sizeof (struct time));
    gotoxy (1,2);
    cprintf ("-* Getting Date...");
    stat2 = get_data (data);
    if (stat2) memcpy (&d,data,sizeof (struct date));
    clrscr ();
    if ( (stat1) && (stat2) ) cprintf ("-* Got Date/Time ");
    if ( (stat1) && (stat2) ) {
        settime (&t);
    }
}

```


SERVER:C

```

                                setdate (&d);
                                put_life ();
                            }
                            return;
    }

/*-----
archive_database
-----*/
archive_database ()
{
    system ("archive o"); /* run archive utility with overnight*/
}

/*-----
--
check_args
-----
-*/
/****
check_args (int n,char *l[])
{
    if (n != 2) {
        main_wt = windowopen (&main_win);
        setttitle (main_wt,"How 'SERVER' Works",CenterUpperTitle);
        clrscr ();
        printf ("SERVER V1.55");
        gotoxy (1,2);
        printf ("    -* Server For GVN Network");
        gotoxy (1,4);
        printf ("USAGE:  server [PORT]");
        gotoxy (1,6);
        printf ("\tRequired:");
        gotoxy (1,7);
        printf ("\t\t[PORT] - 1 .. 4  (COM1 to COM4)");
        gotoxy (1,10);
        printf ("GMM 1990");
        window (1,1,80,25);
        gotoxy (1,24);
        exit (0);
    }
}
****/

/*-----
error :
-----*
/
error (int e)
{
    char message [80];

    sprintf (message,"ERROR  %d",e);
    switch (e) {

```


SERVER.C

```

        case -2 : sprintf (message,"Invalid Port! %d",e);
                    break;
        case -3 : sprintf (message,"Port Already Inuse! %d",e);
                    break;
        case -4 : sprintf (message,"Invalid Buffer Size! %d",e);
                    break;
        case -5 : sprintf (message,"Memory Allocation Error In
Port Setup! %d,e");
                    break;
        case -6 : sprintf (message,"Port Not Setup! %d",e);
                    break;
        case -7 : sprintf (message,"Invalid Parameter! %d",e);
                    break;
        case -23 : sprintf (message,"Modem Not Responding! %d",e);
                    break;
        case -22 : sprintf (message,"Modem Not Responding! %d",e);
                    break;
        case -100: sprintf (message,"Can't Reset Modem! %d",e);
                    break;
    }
    errrtn (message);
    hang_up ();
}

/*-----
-
hang_up
-----
*/
void hang_up ()
{
    int i;
    use (comm_wt);
    clrscr ();
    gotoxy (15,2);
    cprintf ("GVN UnLoading -> ");
    textbackground (Black);
    gotoxy (30,2);
    cprintf ("");
    use (comm_wt); /* resets color etc.. */
    gotoxy (30,2);
    textcolor (Red);
    while (!istxempty (PORT) );
    cprintf ("%c%c",219,219);
    timer (TICKS_PER_SECOND * 1);
    cprintf ("%c%c",219,219);
    asiputs (PORT,"+++",-1);
    cprintf ("%c%c",219,219);
    while (!istxempty (PORT) );
    timer (TICKS_PER_SECOND * 2);
    cprintf ("%c%c",219,219);
    HMSetHookSwitch (PORT, ONHOOK);
    asiquit (PORT);
    cprintf ("%c%c",219,219);

```


SERVER.C

```

    use (comm_wt); /* reset color etc...*/
}

/*-----*/
open_port:
/*-----*/
open_port ()
{
    int stat;

    stat = ASSUCCESS;
    if ((stat = asifirst (PORT,MODE,RXLEN,TXLEN)) < ASSUCCESS){
        error (stat);
    }
    if ((stat = asiinit(PORT,BAUD,PARITY,STOP_BITS,WORD_LENGTH))
        < ASSUCCESS ) {
        error (stat);
    }
    if ( (stat = asdtr(PORT,ON)) < ASSUCCESS)
        error (stat);
    if ( (stat = asrts (PORT,ON)) < ASSUCCESS)
        error (stat);
    if ( (stat = asistart(PORT,ASINOUT)) < ASSUCCESS)
        error (stat);
    HMWaitForOK (TICKS_PER_SECOND*3,NULL); /* wait 3 secs */
    HMSetUpAbortKey (ESC);
}

/*-----*/
init_modem : initialize modem Recursive function
/*-----*/
int init_modem ()
{
    int stat,i;

    i = 0;
    stat = HMReset (PORT); /* reset modem */
    while ( (stat < ASSUCCESS) && (i <= 3) ){
        ++i;
        stat = HMReset (PORT);
        gotoxy (1,3);
        hang_up ();
        open_port();
    }
    if (i > 3) {
        hang_up ();
        open_port ();
        while (init_modem () < 1) {
            errprn ("Couldn't Reset Modem. Contact
Central");
        }
        return ( TRUE );
    }
}

```


SERVER.C

```

                                error (stat);
if (ECHO == 0)
    if ( (stat = HMSetEchoMode (PORT,OFF))
<ASSUCCESS) /* set echo */
                                error (stat);
if (ECHO == 1)
    if ( (stat = HMSetEchoMode (PORT,ON)) <ASSUCCESS)
        error(stat);
if ( (stat = HMSetVerboseMode (PORT,ON)) < ASSUCCESS)
    error (stat);
/* verbal response */
if ( (stat = HMSetFullDuplexMode (PORT,ON)) < ASSUCCESS)/*
duplex FULL */
    error (stat);
if ( (stat = HMSetSpeaker (PORT,SPEAKER)) <ASSUCCESS) /*
set speaker */
    error (stat);

if (i>3)
    return FALSE;

return TRUE;
}

```


SERVER.C

```
/*-----  
-----
```

server For GVN Network

PURPOSE:

Waits for a host to log on and preforms functions.

Written By: Greg McGregor 1990

REVISED:

What was revised?

GMM 7-30-1991	Nothing
GMM 8-13-1991	Started the delete execute commands on TAU
GMM 8-14-1991	Finished Delete and Execute commands on TAU V1.50
GMM 8-26-1991	Adjusted version numbers to 1.52
GMM 9-9-1991	Won't hang on initializing modem

```
-----  
-*/
```

```
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <process.h>  
#include <time.h>  
#include <window.h>  
#include <math.h>  
#include <float.h>  
#include <dos.h>  
#include <bios.h>  
#include <mem.h>  
#include <fcntl.h>  
#include <sys\stat.h>  
#include <io.h>  
#include "asiports.h"  
#include "xfer.h"  
#include "ibmkeys.h"  
#include "gf.h"
```

```
#include <windows.h>  
#include <misc.h>  
#include <time.h>  
#include <gbase.h>  
#include <extnvar.h>
```

```
void status_routine (char *m); /* local commands */  
void transfer_status (XFER *b);  
char calc_CRC (char *s,int len);  
int get_xchar ();  
int set_answer ();  
struct tm far *get_life ();  
void start_server ();  
void end_server ();  
void run_server ();  
void hang_up ();
```


SERVER.C

```
void hang_up1 ();
int is_ring ();
int init_modem ();
```

```
/*
 * Window Defs
 */
windef comm_win =
{10,12,70,17,White,Blue,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
  White,Blue};
windef wait_win = {10,4,70,6,White,Red,FALSE,FALSE,FALSE,TRUE,SINGLEFR
AME,
  White,Red};
windef status_win =
{5,20,75,22,White,Blue,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
  White,Blue};
```

```
/*
 * Window Types
 */
wintype comm_wt,wait_wt,status_wt;
```

```
#define FULL 1
#define HALF 0
#define MODE ASINOUT|BINARY|NORMALRX
#define RXLEN 1024
#define TXLEN 1024
#define SECONDS 5
#define TRUE 1
#define FALSE 0
#define ECHO 0
#define SPEAKER OFF
```

```
char ACK_CHAR = 0x20;
char NAK_CHAR = 0x21;
char LOG_OUT = 0x22;
char SEND_COMMAND = 0x23;
```

```
int PORT;
int BAUD = 2400;          /* Hotels are all at 2400 Baud */
int PARITY = P_NONE;     /* No Parity */
char PHONE_NUMBER [20];  /* phone number to call */
int STOP_BITS = 1;
int WORD_LENGTH = 8;
int DUPLEX = FULL;
char command_list[80];
char file_name[80];
```

```
/*-----
```


SERVER.C

```

--
main:
-----
-----*/
/**
main (int argc, char *argv[])
{
int done;
    done = FALSE;
    init_windows ();
    check_args (argc,argv);
    comm_wt = windowopen (&comm_win);
    setttitle (comm_wt,"GVN Server V1.55",CenterUpperTitle);
    PORT = atoi (argv[1]) - 1;
    while (!done) {
        asiclear (PORT,ASINOUT);
        open_port ();
        init_modem ();
        if (set_answer()) {
            wait_for_commands ();
        } else done = TRUE;
        hang_up ();
    }
}
*****/
/*-----
set_gvn_port
-----*/
set_gvn_port (int port)
{
    PORT = port;
}

/*-----
start_server
-----*/
void start_server ()
{
    comm_wt = windowopen (&comm_win);
    setttitle (comm_wt,"GVN Server V1.55",CenterUpperTitle);
    clrscr ();
    gotoxy (15,2);
    cprintf ("GVN Loading -> ");
    textbackground (Black);
    gotoxy (30,2);
    cprintf ("          ");
    gotoxy (30,2);
    open_port ();
    cprintf ("%c%c",219,219);
    init_modem ();
    cprintf ("%c%c",219,219);
    HMSetWaitTimeForCarrier (PORT,30);
    cprintf ("%c%c",219,219);
    HMSetAutoAnswerRingCount (PORT,1);
}

```


SERVER.C

```

    cprintf ("%c%c",219,219);
    windowclose (comm_wt);
}

/*-----
end_server
-----*/
void end_server ()
{
    comm_wt = windowopen (&comm_win);
    setttitle (comm_wt,"WAIT!",CenterUpperTitle);
    hang_up ();
    windowclose (comm_wt);
}

/*
/*-----
end_server1
-----*/
void end_server1 ()
{
    comm_wt = windowopen (&comm_win);
    setttitle (comm_wt,"WAIT!",CenterUpperTitle);
    hang_up1 ();
    windowclose (comm_wt);
}
*/

/*-----
is_ring
-----*/
int is_ring ()
{
    return (iscd (PORT,IMMEDIATE));
}

/*-----
far run_server
-----*/
void run_server ()
{
    wait_wt = windowopen (&wait_win);
    setttitle (wait_wt,"GVN Network Active",CenterUpperTitle);
    clrscr ();
    textcolor (White+Blink);
    cprintf ("                Please Wait Until Host Is Finished!");
    textcolor (White);
    comm_wt = windowopen (&comm_win);
    setttitle (comm_wt,"GVN Server V1.55",CenterUpperTitle);
/*    HMSetCarrier (PORT,ON); */
    HMAnswer (PORT);
    HMSetHookSwitch (PORT,OFFHOOK);
    if (!wait_for_commands ()) {

```


SERVER.C

```

        use (wait_wt);
        windowclose (wait_wt);
        use (comm_wt);
        windowclose (comm_wt);
    }
}

/*-----
set_answer
-----*/

/
int set_answer ()
{
    int count;
    char ch;

    use (comm_wt);
    clrscr ();
    cprintf("  /*  Waiting For TELEMAT Host Connect!");
    HMSetEscapeCode (PORT,ESC);
    HMSetWaitTimeForCarrier (PORT,30);
    while (HMGetIncomingRingCount (PORT) < 1) {
        if (kbhit ()) {
            ch = getch ();
            if (ch == 0x1B) /*  an ESC key */
                return FALSE;
        }
    }
    HMSetCarrier (PORT,ON);
    HMAnswer (PORT);
    HMSetHookSwitch (PORT,OFFHOOK);
    return TRUE;
}

/*-----
connected : PREDICATE is connected to site
-----*/
int connected ()
{
    return iscd (PORT,CUMULATIVE);
}

/*-----
recieve_command
-----*/
char recieve_command (char c)
{
    int stat;
    int trys = 0;

    while ( ( (stat = get_xchar ()) != c) && (trys < 100) ) {
        ++ trys;
        send_xchar (NAK_CHAR);
    }
}

```


SERVER.C

```

    }
    if (stat == c ) {
        send_xchar (ACK_CHAR);
        return TRUE;
    }
    return FALSE; /* error */
}

```

```

/*-----
send_command
-----*/

```

```

char send_command (char c)
{
    int stat;
    int trys;

    trys = 0;
    send_xchar (c);
    do {
        stat = get_xchar ();
        if (stat != 0) cprintf ("%g",stat);
        ++trys;
        send_xchar (c);
    } while ( (stat != ACK_CHAR) && (trys < 30) );

    if (stat == ACK_CHAR)
        return TRUE;
    return FALSE;
}

```

```

/*-----
wait_for_commands
-----*/

```

```

int wait_for_commands ()
{
    int stat,a_command,done,B_connected,trys;
    int idle_time = 0;
    int fd,HOST_LOCKED_MODE = FALSE;
    char s[20];

    B_connected = FALSE;
    done = FALSE;
    trys = 0;
    while (!connected ()) ;
    asiclear (PORT,ASINOUT);
    while (!done) {
        use (comm_wt);
        clrscr ();
        cprintf ("-* Sending Job Request [ ]");
        trys = 0;
        while (!send_command (SEND_COMMAND) ) {
            clrscr ();
            ++trys;

```


SERVER.C

```

                                                                    134
cprintf ("-* Sending Job Request [%d]", trys);
if (trys >= 3) return;
}
clrscr ();
cprintf ("-* Waiting For A Command");
stat = get_xchar (); /* wait for a command */
if (stat == 0) ++idle_time;
if (stat == 0x02) { /* request for file transfer*/
    clrscr ();
    cprintf ("-* Host Requesting An UPLINK!");
    send_xchar (ACK_CHAR);
    file_send();
    idle_time = 0;
} else
if (stat == 0x01) {
    clrscr ();
    cprintf ("-* Host DOWNLINKING A File");
    timer (TICKS_PER_SECOND);
    send_xchar (ACK_CHAR);
    file_receive();
    idle_time = 0;
} else
if (stat == 0x03) {
    clrscr ();
    cprintf ("-* Host Requested An Archive!");
    send_xchar (ACK_CHAR);
    archive_database ();
    idle_time = 0;
} else
if (stat == 0x05) {
    clrscr ();
    cprintf ("-* Date/Time Verification...");
    send_xchar (ACK_CHAR);
    date_time_set ();
    idle_time = 0;
} else
if (stat == 0x08) {
    clrscr ();
    cprintf ("-* Sending Agreement Records To Host");
    send_xchar (ACK_CHAR);
    send_agreemnt();
    idle_time = 0;
} else
if (stat == 0x09) {
    clrscr ();
    cprintf ("-* Sending Phone File To Host");
    send_xchar (ACK_CHAR);
    send_phone ();
    idle_time = 0;
} else
if (stat == 0x0A) {
    clrscr ();
    cprintf ("-* Host Requesting A Reboot!");
    send_xchar (ACK_CHAR);

```


SERVER.C

```

reboot ();
} else
if (stat == 0x08) {
    clrscr ();
    cprintf ("-* Host Locking Software!");
    send_xchar (ACK_CHAR);
    end_server ();
    lock_software ();
    idle_time = 0;
    done = TRUE;
    HOST_LOCKED_MODE = TRUE;
} else
if (stat == 0x0C) {
    clrscr ();
    cprintf ("-* Host Requesting Serial Number!");
    send_xchar (ACK_CHAR);
    send_serial_number ();
    idle_time = 0;
} else
if (stat == 0x0D) {
    clrscr ();
    cprintf ("-* Receiving Monthly Vitamins!");
    send_xchar (ACK_CHAR);
    put_life ();
    idle_time = 0;
} else
if (stat == 0x0E) {
    clrscr ();
    cprintf ("-* Host Unlocking Software!");
    send_xchar (ACK_CHAR);
    SYSTEM_LOCKED = FALSE;
    fd = open ("c:\\\\~\\\\~", O_WRONLY|O_BINARY|O_TRUNC, S_IWRITE);
    s[0] = '1'; /* reset file flag as unlocked */
    write (fd, s, 1);
    close (fd);
    idle_time = 0;
    done = TRUE;
} else
if (stat == 0x0F) {
    clrscr ();
    cprintf ("-* Host Requesting A File ZAP!");
    send_xchar (ACK_CHAR);
    zap_file ();
    idle_time = 0;
} else
if (stat == 0x10) {
    clrscr ();
    cprintf ("-* Host Requesting A File EXECUTE!");
    send_xchar (ACK_CHAR);
    execute_file ();
    idle_time = 0;
} else
if (stat == 0x11) {
    clrscr ();

```


SERVER.C

```

                                cprintf ("-* Host Requesting A Data LOCK!");
                                send_xchar (ACK_CHAR);
                                ME_LOCK = TRUE;
        } else
        if (stat == 0x12) {
                                clrscr ();
                                cprintf ("-* Host Requesting A Data UNLOCK!");
                                send_xchar (ACK_CHAR);
                                ME_LOCK = FALSE;
        } else
        if (stat == LOG_OUT){
                                gotoxy (1,2);
                                cprintf ("Log out!");
                                done = TRUE;
        } else
                                send_xchar (NAK_CHAR);

        if (idle_time > 20) {
                                cprintf ("Timed out!");
                                return; /* no command for 21 loops */
        }
    }
    return HOST_LOCKED_MODE;
}

/*-----
get_xchar: get char from line
-----*/
int get_xchar ()
{
    int stat;
    int trys = 0;

    while ( (stat = asigetc (PORT)) < ASSUCCESS) && (trys < 10000) ){
        ++trys;
    }
    if (trys < 10000) return stat;
    return 0;
}

/*-----
send_xchar : send a char down line
-----*/
send_xchar (char c)
{
    int stat;
    while ( (stat = asiputc (PORT,c)) < ASSUCCESS) ;
}

/*-----
get_data
-----*/

```


SERVER.C

```

int get_data (char *data)
{
char bytes,crc,crc1;
int i,j,k,stat,return_value;
char temp[256],temp1[256];

    k = 3;
    while (k) {
        while ( (bytes = get_xchar ()) == 0);
        use (comm_wt);
        clrscr ();
        cprintf ("-* Bytes Coming %d ",bytes);
        i = 0;
        while (i<bytes-1) {
            temp[i] = get_xchar ();
            ++i;
        }

        crc = temp[i-1];
        temp[i-1] = '\0';
        crc1 = calc_CRC (temp,(bytes - 2));
        if (crc != crc1) {
            return_value = FALSE;
            --k;
            asiputc (PORT,NAK_CHAR);
            cprintf ("-* Retrying Data...");
        }
        if (crc == crc1 ) {
            k =0;
            return_value = TRUE;
            asiputc (PORT,ACK_CHAR);
        }
    }
    strncpy (data,temp,bytes-2);
    data[bytes-2] = '\0';
    return return_value;
}

/*-----
calc_CRC
-----*/
char calc_CRC (char *s,int len)
{
int i,j;
char crc;

    crc = 0;
    i = len;
    crc = s[0];
    for (j=1;j<i;j++)
        crc = crc ^ s[j];
    return crc;
}

```


SERVER.C

```

/*-----
file_receive
-----*/
file_receive ()
{
    int stat;
        stat = YmodemReceive (PORT,status_routine,NULL,ESC);
        gotoxy (1,4);
        cprintf ("File Transfer Status %d",stat);
}

/*-----
send_agreemnt
-----*/
send_agreemnt ()
{
    int stat;
        stat = YmodemSend (PORT,"agreemnt.",status_routine,NULL,ESC);
}

/*-----
send_phone
-----*/
send_phone ()
{
    int stat;
        stat = YmodemSend (PORT,"phone.",status_routine,NULL,ESC);
}

/*-----
file_send
-----*/
file_send ()
{
    int stat;
    char file_name [80];

        cprintf ("-* Data Coming...");
        if (!get_data (file_name)) {
            clrscr ();
            cprintf ("-* Couldn't Get File Name From Host");
        } else {
            clrscr ();
            cprintf ("Host Requested File '%s'",file_name);
            timer (TICKS_PER_SECOND * 3); /* wait 3 seconds before
send starts */
            stat = YmodemSend (PORT,file_name,status_routine,NULL,ESC);
        }
}

void status_routine (char *m)
{

```


SERVER.C

```

gotoxy (1,3);
cprintf ("");
gotoxy (1,3);
cprintf ("%s\n",m);
}

/*-----
reboot ()
-----*/
reboot () {
    system ("reboot");
}

/*-----
send_serial_number
-----*/
send_serial_number () {
    int fd;
    int stat;
    fd = open
("serial.dat",O_WRONLY|O_TEXT|O_TRUNC|O_CREAT,S_IREAD|S_IWRITE);
    write (fd,"Not implemented!",strlen ("Not implemented!"));
    close (fd);
    stat = YmodemSend (PORT,"serial.dat",status_routine,NULL,ESC);
}

/*-----
lock_software
-----*/
lock_software () {
    end_server ();
    lock_system ();
}

/*-----
file_exists
-----*/
int file_exists (char *s)
{
    FILE *f;
    f = fopen (s,"r");
    if (f == NULL)
        return FALSE;
    fclose (f);
    return TRUE;
}

/*-----
put_life: re-birth software
-----*/

```


SERVER.C

```

put_life ()
{
    struct tm *t;
    time_t tm;
    int fd;

    tm = time(NULL);
    t = localtime (&tm);
    t->tm_yday += 30; /* add one month to life span */
    if (t->tm_yday > 365) { /* allow for year change */
        t->tm_year++;
        t->tm_yday = t->tm_yday - 365;
    }
    fd = open ("lspan.dat", O_BINARY|O_WRONLY|O_CREAT|O_TRUNC, S_IWRITE);
    if (fd != -1) {
        write (fd, t, sizeof (struct tm));
    }
    close (fd);
}

```

```

/*-----*/
get_life : return Birthday of software
/*-----*/

```

```

struct tm life; /* global */

```

```

struct tm *get_life ()
{
    int fd;

    if (!file_exists ("lspan.dat")) {
        return NULL;
    }
    fd = open ("lspan.dat", O_BINARY|O_RDONLY, S_IREAD);
    if (fd == -1) {
        return NULL;
    }
    read (fd, &life, sizeof (struct tm));
    close (fd);
    return &life;
}

```

```

/*-----*/
zap_file
/*-----*/

```

```

zap_file () {
    int stat;
    char file_name [80];
    char cmd_string [255];
}

```


SERVER.C

```

if (!get_data (file_name)) {
    clrscr ();
    cprintf ("--* Couldn't Get File Name From Host");
} else {
    strcpy (command_string,"del ");
    strcat (command_string,file_name);
    system (command_string);
}

```

```

/*-----
execute_file
-----*/

```

```

execute_file () {
    int stat;
    char file_name [80];

    gotoxy (1,2);
    cprintf ("--* Data Coming...");
    if (!get_data (file_name)) {
        clrscr ();
        cprintf ("--* Couldn't Get File Name From Host");
    } else {
        system (file_name);
    }
}

```

```

/*-----
date_time_set () : set date and time according to host
-----*
/

```

```

date_time_set ()
{
    struct time t;
    struct date d;
    int stat1,stat2;
    char data[80];

    use (comm_wt);
    clrscr ();
    cprintf ("--* Getting Time...");
    stat1 = get_data (data);
    if (stat1) memcpy (&t,data,sizeof (struct time));
    gotoxy (1,2);
    cprintf ("--* Getting Date...");
    stat2 = get_data (data);
    if (stat2) memcpy (&d,data,sizeof (struct date));
    clrscr ();
    if ( (stat1) && (stat2) ) cprintf ("--* Got Date/Time ");
    if ( (stat1) && (stat2) ) {
        settime (&t);
    }
}

```


SERVER.C

```

        setdate (&d);
        put_life ();
    }
    return;
}

/*-----
archive_database
-----*/
archive_database ()
{
    system ("archive o"); /* run archive utility with overnight*/
}

/*-----
---
check_args
-----
-*/
/**
check_args (int n,char *l[])
{
    if (n != 2) {
        main_wt = windowopen (&main_win);
        setttitle (main_wt,"How 'SERVER' Works",CenterUpperTitle);
        clrscr ();
        printf ("SERVER V1.55");
        gotoxy (1,2);
        printf ("    -* Server For GVN Network");
        gotoxy (1,4);
        printf ("USAGE:  server [PORT]");
        gotoxy (1,6);
        printf ("\tRequired:");
        gotoxy (1,7);
        printf ("\t\t[PORT] = 1 .. 4  (COM1 to COM4)");
        gotoxy (1,10);
        printf ("GMM 1990");
        window (1,1,80,25);
        gotoxy (1,24);
        exit (0);
    }
}
****/

/*-----
error :
-----*
/
error (int e)
{
    char message [80];

    sprintf (message,"ERROR  %d",e);
    switch (e) {

```


SERVER.C

```

        case -2 : sprintf (message,"Invalid Port! %d",e);
                    break;
        case -3 : sprintf (message,"Port Already Inuse! %d",e);
                    break;
        case -4 : sprintf (message,"Invalid Buffer Size! %d",e);
                    break;
        case -5 : sprintf (message,"Memory Allocation Error In
Port Setup! %d,e");
                    break;
        case -6 : sprintf (message,"Port Not Setup! %d",e);
                    break;
        case -7 : sprintf (message,"Invalid Parameter! %d",e);
                    break;
        case -23 : sprintf (message,"Modem Not Responding! %d",e);
                    break;
        case -22 : sprintf (message,"Modem Not Responding! %d",e);
                    break;
        case -100: sprintf (message,"Can't Reset Modem! %d",e);
                    break;
    }
    errrtn (message);
    hang_up ();
}

```

```

/*-----
-
hang_up
-----
*/

```

```

void hang_up ()
{
    int i;

    use (comm_wt);
    clrscr ();
    gotoxy (15,2);
    cprintf ("GVN UnLoading -> ");
    textbackground (Black);
    gotoxy (30,2);
    cprintf ("");
    use (comm_wt); /* resets color etc.. */
    gotoxy (30,2);
    textcolor (Red);
    while (!istxempty (PORT) );
    cprintf ("%c%c",219,219);
    timer (TICKS_PER_SECOND * 1);
    cprintf ("%c%c",219,219);
    asiputs (PORT,"+++",-1);
    cprintf ("%c%c",219,219);
    while (!istxempty (PORT) );
    timer (TICKS_PER_SECOND * 2);
    cprintf ("%c%c",219,219);
    HMSetHookSwitch (PORT, ONHOOK);
    asiquit (PORT);
    cprintf ("%c%c",219,219);
}

```


SERVER.C

```

use (comm_wt); /* reset color etc...*/
}

/*-----
open_port:
-----*/
open_port ()
{
int stat;

stat = ASSUCCESS;
if ((stat = asifirst (PORT,MODE,RXLEN,TXLEN)) < ASSUCCESS){
    error (stat);
}
if ((stat = asiinit(PORT,BAUD,PARITY,STOP_BITS,WORD_LENGTH))
    < ASSUCCESS ) {
    error (stat);
}
if ( (stat = asdtr(PORT,ON)) < ASSUCCESS)
    error (stat);
if ( (stat = asrts (PORT,ON)) < ASSUCCESS)
    error (stat);
if ( (stat = asistart(PORT,ASINOUT)) < ASSUCCESS)
    error (stat);
HMWaitForOK (TICKS_PER_SECOND*3,NULL); /* wait 3 secs */
HMSetUpAbortKey (ESC);
}

/*-----
init_modem : initialize modem Recursive function
-----*/
int init_modem ()
{
int stat,i;

i = 0;
stat = HMReset (PORT); /* reset modem */
while ( (stat < ASSUCCESS) && (i <= 3) ){
    ++i;
    stat = HMReset (PORT);
    gotoxy (1,3);
    hang_up ();
    open_port();
}
if (i > 3) {
    hang_up ();
    open_port ();
    while (init_modem () < 1) {
        errrtn ("Couldn't Reset Modem. Contact
Central");
    } return ( TRUE );
}

if (stat < ASSUCCESS)

```


SERVER.C

```
        error (stat);
    if (ECHO == 0)
        if ( (stat = HMSetEchoMode (PORT,OFF))
<ASSUCCESS) /* set echo */
            error (stat);
    if (ECHO == 1)
        if ( (stat = HMSetEchoMode (PORT,ON)) <ASSUCCESS)
            error(stat);
    if ( (stat = HMSetVerboseMode (PORT,ON)) < ASSUCCESS)
        error (stat);
        /* verbal response */
    if ( (stat = HMSetFullDuplexMode (PORT,ON)) < ASSUCCESS)/*
duplex FULL */
        error (stat);
    if ( (stat = HMSetSpeaker (PORT,SPEAKER)) <ASSUCCESS) /*
set speaker */
        error (stat);

    if (i>3)
        return FALSE;

    return TRUE;
}
```


SERVER.C

```
/*-----
server  For GVN Network
```

```
PURPOSE:
    Waits for a host to log on and preforms functions.
```

```
Written By: Greg McGregor 1990
```

```
REVISED:          What was revised?
```

```
GMM 7-30-1991      Nothing
GMM 8-13-1991      Started the delete execute commands on TAU
GMM 8-14-1991      Finished Delete and Execute commands on TAU V1.50
GMM 8-26-1991      Adjusted version numbers to 1.52
GMM 9-9-1991       Won't hang on initializing modem
```

```
*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <process.h>
#include <time.h>
#include <window.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <mem.h>
#include <fcntl.h>
#include <sys\stat.h>
#include <io.h>
#include "asiports.h"
#include "xfer.h"
#include "ibmkeys.h"
#include "gf.h"
```

```
#include <windows.h>
#include <misc.h>
#include <time.h>
#include <gbase.h>
#include <extnvar.h>
```

```
void status_routine (char *m);    /* local commands */
void transfer_status (XFER *b);
char calc_CRC (char *s,int len);
int get_xchar ();
int set_answer ();
struct tm far *get_life ();
void start_server ();
void end_server ();
void run_server ();
void hang_up ();
```


SERVER.C

```
void hang_up1 ();
int is_ring ();
int init_modem ();

/*
 * Window Defs
 */
windef comm_win =
{10,12,70,17,White,Blue,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
                                White,Blue};
windef wait_win = {10,4,70,6,White,Red,FALSE,FALSE,FALSE,TRUE,SINGLEFR
AME,
                                White,Red};
windef status_win =
{5,20,75,22,White,Blue,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
                                White,Blue};

/*
 * Window Types
 */
wintype comm_wt,wait_wt,status_wt;

#define FULL 1
#define HALF 0
#define MODE ASINOUT|BINARY|NORMALRX
#define RXLEN 1024
#define TXLEN 1024
#define SECONDS 5
#define TRUE 1
#define FALSE 0
#define ECHO 0
#define SPEAKER OFF

char ACK_CHAR = 0x20;
char NAK_CHAR = 0x21;
char LOG_OUT = 0x22;
char SEND_COMMAND = 0x23;

int PORT;
int BAUD = 2400;
int PARITY = P_NONE;
char PHONE_NUMBER [20];
int STOP_BITS = 1;
int WORD_LENGTH = 8;
int DUPLEX = FULL;
char command_list[80];
char file_name[80];

/* Hotels are all at 2400 Baud */
/* No Parity */
/* phone number to call */
```

SERVER.C

```

--
main:
-----
--*/
/**
main (int argc, char *argv[])
{
    int done;
        done = FALSE;
        init_windows ();
        check_args (argc,argv);
        comm_wt = windowopen (&comm_win);
        setttitle (comm_wt,"GVN Server V1.55",CenterUpperTitle);
        PORT = atoi (argv[1]) - 1;
    while (!done) {
        asiclear (PORT,ASINOUT);
        open_port ();
        init_modem ();
        if (set_answer()) {
            wait_for_commands ();
        } else done = TRUE;
        hang_up ();
    }
}
*****/
/*-----
set_gvn_port
-----*/
set_gvn_port (int port)
{
    PORT = port;
}

/*-----
start_server
-----*/
void start_server ()
{
    comm_wt = windowopen (&comm_win);
    setttitle (comm_wt,"GVN Server V1.55",CenterUpperTitle);
    clrscr ();
    gotoxy (15,2);
    cprintf ("GVN Loading -> ");
    textbackground (Black);
    gotoxy (30,2);
    cprintf ("          ");
    gotoxy (30,2);
    open_port ();
    cprintf ("%c%c",219,219);
    init_modem ();
    cprintf ("%c%c",219,219);
    HMSetWaitTimeForCarrier (PORT,30);
    cprintf ("%c%c",219,219);
    HMSetAutoAnswerRingCount (PORT,1);
}

```


SERVER.C

```

    cprintf ("%c%c",219,219);
    windowclose (comm_wt);
}

/*-----
end_server
-----*/
void end_server ()
{
    comm_wt = windowopen (&comm_win);
    setttitle (comm_wt,"WAIT!",CenterUpperTitle);
    hang_up ();
    windowclose (comm_wt);
}

/*
/*-----
end_server1
-----*/
void end_server1 ()
{
    comm_wt = windowopen (&comm_win);
    setttitle (comm_wt,"WAIT!",CenterUpperTitle);
    hang_up1 ();
    windowclose (comm_wt);
}
*/

/*-----
is_ring
-----*/
int is_ring ()
{
    return (iscd (PORT,IMMEDIATE));
}

/*-----
far run_server
-----*/
void run_server ()
{
    wait_wt = windowopen (&wait_win);
    setttitle (wait_wt,"GVN Network Active",CenterUpperTitle);
    clrscr ();
    textcolor (White+Blink);
    cprintf ("          Please Wait Until Host Is Finished!");
    textcolor (White);
    comm_wt = windowopen (&comm_win);
    setttitle (comm_wt,"GVN Server V1.55",CenterUpperTitle);
/* HMSetCarrier (PORT,ON); */
    HMAnswer (PORT);
    HMSetHookSwitch (PORT,OFFHOOK);
    if (!wait_for_commands ()) {

```


SERVER.C

```

        use (wait_wt);
        windowclose (wait_wt);
        use (comm_wt);
        windowclose (comm_wt);
    }
}

/*-----*/
set_answer
/*-----*/

/
int set_answer ()
{
    int count;
    char ch;

    use (comm_wt);
    clrscr ();
    cprintf("  /* Waiting For TELEMAT Host Connect!");
    HMSetEscapeCode (PORT,ESC);
    HMSetWaitTimeForCarrier (PORT,30);
    while (HMGetIncomingRingCount (PORT) < 1) {
        if (kbhit ()) {
            ch = getch ();
            if (ch == 0x1B) /* an ESC key */
                return FALSE;
        }
    }
    HMSetCarrier (PORT,ON);
    HMAnswer (PORT);
    HMSetHookSwitch (PORT,OFFHOOK);
    return TRUE;
}

/*-----*/
connected : PREDICATE is connected to site
/*-----*/

int connected ()
{
    return iscd (PORT,CUMULATIVE);
}

/*-----*/
recieve_command
/*-----*/

char recieve_command (char c)
{
    int stat;
    int trys = 0;

    while ( ( (stat = get_xchar ()) != c) && (trys < 100) ) {
        trys++;
    }
}

```


SERVER.C

```

    }
    if (stat == c ) {
        send_xchar (ACK_CHAR);
        return TRUE;
    }
    return FALSE; /* error */
}

```

```

/*-----
send_command
-----*/

```

```

char send_command (char c)
{
    int stat;
    int trys;

    trys = 0;
    send_xchar (c);
    do {
        stat = get_xchar ();
        if (stat != 0) cprintf ("%g",stat);
        ++trys;
        send_xchar (c);
    } while ( (stat != ACK_CHAR) && (trys < 30) );

    if (stat == ACK_CHAR)
        return TRUE;
    return FALSE;
}

```

```

/*-----
wait_for_commands
-----*/

```

```

int wait_for_commands ()
{
    int stat,a_command,done,B_connected,trys;
    int idle_time = 0;
    int fd,HOST_LOCKED_MODE = FALSE;
    char s[20];
    B_connected = FALSE;
    done = FALSE;
    trys = 0;
    while (!connected ()) ;
    asiclear (PORT,ASINOUT);
    while (!done) {
        use (comm_wt);
        clrscr ();
        cprintf ("-* Sending Job Request [ ]");
        trys = 0;
        while (!send_command (SEND_COMMAND) ) {
            clrscr ();
            ++trys;

```


SERVER.C

```
        cprintf ("-* Sending Job Request [%d]", trys);
        if (trys >= 3) return;
    }
    clrscr ();
    cprintf ("-* Waiting For A Command");
    stat = get_xchar (); /* wait for a command */
    if (stat == 0) ++idle_time;
    if (stat == 0x02) { /* request for file transfer */
        clrscr ();
        cprintf ("-* Host Requesting An UPLINK!");
        send_xchar (ACK_CHAR);
        file_send();
        idle_time = 0;
    } else
    if (stat == 0x01) {
        clrscr ();
        cprintf ("-* Host DOWNLINKING A File");
        timer (TICKS_PER_SECOND);
        send_xchar (ACK_CHAR);
        file_receive();
        idle_time = 0;
    } else
    if (stat == 0x03) {
        clrscr ();
        cprintf ("-* Host Requested An Archive!");
        send_xchar (ACK_CHAR);
        archive_database ();
        idle_time = 0;
    } else
    if (stat == 0x05) {
        clrscr ();
        cprintf ("-* Date/Time Verification...");
        send_xchar (ACK_CHAR);
        date_time_set ();
        idle_time = 0;
    } else
    if (stat == 0x08) {
        clrscr ();
        cprintf ("-* Sending Agreement Records To Host");
        send_xchar (ACK_CHAR);
        send_agreemnt();
        idle_time = 0;
    } else
    if (stat == 0x09) {
        clrscr ();
        cprintf ("-* Sending Phone File To Host");
        send_xchar (ACK_CHAR);
        send_phone ();
        idle_time = 0;
    } else
    if (stat == 0x0A) {
        clrscr ();
        cprintf ("-* Host Requesting A Reboot!");
        send_xchar (ACK_CHAR);
```


SERVER.C

```
        reboot ();
    } else
    if (stat == 0x08) {
        clrscr ();
        cprintf ("-* Host Locking Software!");
        send_xchar (ACK_CHAR);
        end_server ();
        lock_software ();
        idle_time = 0;
        done = TRUE;
        HOST_LOCKED_MODE = TRUE;
    } else
    if (stat == 0x0C) {
        clrscr ();
        cprintf ("-* Host Requesting Serial Number!");
        send_xchar (ACK_CHAR);
        send_serial_number ();
        idle_time = 0;
    } else
    if (stat == 0x0D) {
        clrscr ();
        cprintf ("-* Receiving Monthly Vitamins!");
        send_xchar (ACK_CHAR);
        put_life ();
        idle_time = 0;
    } else
    if (stat == 0x0E) {
        clrscr ();
        cprintf ("-* Host Unlocking Software!");
        send_xchar (ACK_CHAR);
        SYSTEM_LOCKED = FALSE;
        fd = open ("c:\\\\~\\\\~", O_WRONLY|O_BINARY|O_TRUNC, S_IWRITE);
        s[0] = '1';          /* reset file flag as unlocked */
        write (fd, s, 1);
        close (fd);
        idle_time = 0;
        done = TRUE;
    } else
    if (stat == 0x0F) {
        clrscr ();
        cprintf ("-* Host Requesting A File ZAP!");
        send_xchar (ACK_CHAR);
        zap_file ();
        idle_time = 0;
    } else
    if (stat == 0x10) {
        clrscr ();
        cprintf ("-* Host Requesting A File EXECUTE!");
        send_xchar (ACK_CHAR);
        execute_file ();
        idle_time = 0;
    } else
    if (stat == 0x11) {
        clrscr ();
```


SERVER.C

```

        cprintf ("-* Host Requesting A Data LOCK!");
        send_xchar (ACK_CHAR);
        ME_LOCK = TRUE;
    } else
    if (stat == 0x12) {
        clrscr ();
        cprintf ("-* Host Requesting A Data UNLOCK!");
        send_xchar (ACK_CHAR);
        ME_LOCK = FALSE;
    } else
    if (stat == LOG_OUT){
        gotoxy (1,2);
        cprintf ("Log out!");
        done = TRUE;
    } else
        send_xchar (NAK_CHAR);

    if (idle_time > 20) {
        cprintf ("Timed out!");
        return; /* no command for 21 loops */
    }
}
return HOST_LOCKED_MODE;
}

/*-----
get_xchar: get char from line
-----*/
int get_xchar ()
{
    int stat;
    int trys = 0;

    while ( ( (stat = asigetc (PORT)) < ASSUCCESS) && (trys < 10000) ){
        ++trys;
    }
    if (trys < 10000) return stat;
    return 0;
}

/*-----
send_xchar : send a char down line
-----*/
send_xchar (char c)
{
    int stat;
    while ( (stat = asiputc (PORT,c)) < ASSUCCESS) ;
}

/*-----
get_data
-----*/

```


SERVER.C

```

int get_data (char *data)
{
char bytes,crc,crc1;
int i,j,k,stat,return_value;
char temp[256],temp1[256];

    k = 3;
    while (k) {
        while ( (bytes = get_xchar ()) == 0);
        use (comm_wt);
        clrscr ();
        cprintf ("-* Bytes Coming %d ",bytes);
        i = 0;
        while (i<bytes-1) {
            temp[i] = get_xchar ();
            ++i;
        }

        crc = temp[i-1];
        temp[i-1] = '\0';
        crc1 = calc_CRC (temp,(bytes - 2));
        if (crc != crc1) {
            return_value = FALSE;
            --k;
            asiputc (PORT,NAK_CHAR);
            cprintf ("-* Retrying Data...");
        }
        if (crc == crc1 ) {
            k =0;
            return_value = TRUE;
            asiputc (PORT,ACK_CHAR);
        }
    }
    strncpy (data,temp,bytes-2);
    data[bytes-2] = '\0';
    return return_value;
}

/*-----
calc_CRC
-----*/
char calc_CRC (char *s,int len)
{
int i,j;
char crc;

    crc = 0;
    i = len;
    crc = s[0];
    for (j=1;j<i;j++)
        crc = crc ^ s[j];
    return crc;
}

```


SERVER.C

```

161
/*-----
file_receive
-----*/
file_receive ()
{
    int stat;
        stat = YmodemReceive (PORT,status_routine,NULL,ESC);
        gotoxy (1,4);
        cprintf ("File Transfer Status %d",stat);
}

/*-----
send_agreemnt
-----*/
send_agreemnt ()
{
    int stat;
        stat = YmodemSend (PORT,"agreemnt.",status_routine,NULL,ESC);
}

/*-----
send_phone
-----*/
send_phone ()
{
    int stat;
        stat = YmodemSend (PORT,"phone.",status_routine,NULL,ESC);
}

/*-----
file_send
-----*/
file_send ()
{
    int stat;
    char file_name [80];

        cprintf ("-* Data Coming...");
        if (!get_data (file_name)) {
            clrscr ();
            cprintf ("-* Couldn't Get File Name From Host");
        } else {
            clrscr ();
            cprintf ("Host Requested File '%s'",file_name);
            timer (TICKS_PER_SECOND * 3); /* wait 3 seconds before
send starts */
            stat = YmodemSend (PORT,file_name,status_routine,NULL,ESC);
        }
}

void status_routine (char *m)
{

```


SERVER.C

```

    gotoxy (1,3);
    cprintf ("                ");
    gotoxy (1,3);
    cprintf ("%s\n",m);
}

/*-----
reboot ()
-----*/
reboot () {
    system ("reboot");
}

/*-----
send_serial_number
-----*/
send_serial_number () {
    int fd;
    int stat;
    fd = open
("serial.dat",O_WRONLY|O_TEXT|O_TRUNC|O_CREAT,S_IREAD|S_IWRITE);
    write (fd,"Not implemented!",strlen ("Not implemented!"));
    close (fd);
    stat = YmodemSend (PORT,"serial.dat",status_routine,NULL,ESC);
}

/*-----
lock_software
-----*/
lock_software () {
    end_server ();
    lock_system ();
}

/*-----
file_exists
-----*/
int file_exists (char *s)
{
    FILE *f;
    f = fopen (s,"r");
    if (f == NULL)
        return FALSE;
    fclose (f);
    return TRUE;
}

/*-----
put_life: re-birth software
-----*/

```


SERVER.C

```

put_life ()
{
    struct tm *t;
    time_t tm;
    int fd;

    tm = time(NULL);
    t = localtime (&tm);
    t->tm_yday += 30; /* add one month to life span */
    if (t->tm_yday > 365) { /* allow for year change */
        t->tm_year++;
        t->tm_yday = t->tm_yday - 365;
    }
    fd = open ("lspan.dat", O_BINARY|O_WRONLY|O_CREAT|O_TRUNC, S_IWRITE);
    if (fd != -1) {
        write (fd, t, sizeof (struct tm));
    }
    close (fd);
}

```

```

/*-----
get_life : return Birthday of software
-----*/

```

```

struct tm life; /* global */

struct tm *get_life ()
{
    int fd;

    if (!file_exists ("lspan.dat")) {
        return NULL;
    }
    fd = open ("lspan.dat", O_BINARY|O_RDONLY, S_IREAD);
    if (fd == -1) {
        return NULL;
    }
    read (fd, &life, sizeof (struct tm));
    close (fd);
    return &life;
}

```

```

/*-----
zap_file
-----*/

```

```

zap_file () {
    int stat;
    char file_name [80];
    char command_string[255];

    gotoxy (1,2);
    cprintf ("-* Data Coming...");
}

```


SERVER.C

```

164
if (!get_data (file_name)) {
    clrscr ();
    cprintf ("-* Couldn't Get File Name From Host");
} else {
    strcpy (command_string,"del ");
    strcat (command_string,file_name);
    system (command_string);
}

}

/*-----
execute_file
-----*/
execute_file () {
    int stat;
    char file_name [80];

    gotoxy (1,2);
    cprintf ("-* Data Coming...");
    if (!get_data (file_name)) {
        clrscr ();
        cprintf ("-* Couldn't Get File Name From Host");
    } else {
        system (file_name);
    }
}

/*-----
date_time_set () : set date and time according to host
-----*/
/
date_time_set ()
{
    struct time t;
    struct date d;
    int stat1,stat2;
    char data[80];

    use (comm_wt);
    clrscr ();
    cprintf ("-* Getting Time...");
    stat1 = get_data (data);
    if (stat1) memcpy (&t,data,sizeof (struct time));
    gotoxy (1,2);
    cprintf ("-* Getting Date...");
    stat2 = get_data (data);
    if (stat2) memcpy (&d,data,sizeof (struct date));
    clrscr ();
    if ( (stat1) && (stat2) ) cprintf ("-* Got Date/Time ");
    if ( (stat1) && (stat2) ) {
        settime (&t);
    }
}

```


SERVER.C

```

        setdate (&d);
        put_life ();
    }
    return;
}

/*-----
archive_database
-----*/
archive_database ()
{
    system ("archive o"); /* run archive utility with overnight*/
}

/*-----
--
check_args
-----
-*/
****
check_args (int n,char *l[])
{
    if (n != 2) {
        main_wt = windowopen (&main_win);
        setttitle (main_wt,"How 'SERVER' Works",CenterUpperTitle);
        clrscr ();
        printf ("SERVER V1.55");
        gotoxy (1,2);
        printf ("    -* Server For GVN Network");
        gotoxy (1,4);
        printf ("USAGE:  server [PORT]");
        gotoxy (1,6);
        printf ("\tRequired:");
        gotoxy (1,7);
        printf ("\t\t[PORT] - 1 .. 4  (COM1 to COM4)");
        gotoxy (1,10);
        printf ("GMM 1990");
        window (1,1,80,25);
        gotoxy (1,24);
        exit (0);
    }
}
****/

/*-----
error :
-----*
/
error (int e)
{
    char message [80];

    sprintf (message,"ERROR  %d",e);
    switch (e) {

```


SERVER.C

```

        case -2 : sprintf (message, "Invalid Port! %d", e);
                    break;
        case -3 : sprintf (message, "Port Already Inuse! %d", e);
                    break;
        case -4 : sprintf (message, "Invalid Buffer Size! %d", e);
                    break;
        case -5 : sprintf (message, "Memory Allocation Error In
Port Setup! %d, e");
                    break;
        case -6 : sprintf (message, "Port Not Setup! %d", e);
                    break;
        case -7 : sprintf (message, "Invalid Parameter! %d", e);
                    break;
        case -23 : sprintf (message, "Modem Not Responding! %d", e);
                    break;
        case -22 : sprintf (message, "Modem Not Responding! %d", e);
                    break;
        case -100: sprintf (message, "Can't Reset Modem! %d", e);
                    break;
    }
    errrtn (message);
    hang_up ();
}

```

```

/*-----
-
hang_up
-----
*/

```

```

void hang_up ()
{
    int i;
    use (comm_wt);
    clrscr ();
    gotoxy (15, 2);
    cprintf ("GVN UnLoading -> ");
    textbackground (Black);
    gotoxy (30, 2);
    cprintf ("");
    use (comm_wt); /* resets color etc.. */
    gotoxy (30, 2);
    textcolor (Red);
    while (!listxempty (PORT));
    cprintf ("%c%c", 219, 219);
    timer (TICKS_PER_SECOND * 1);
    cprintf ("%c%c", 219, 219);
    asiputs (PORT, "+++", -1);
    cprintf ("%c%c", 219, 219);
    while (!listxempty (PORT));
    timer (TICKS_PER_SECOND * 2);
    cprintf ("%c%c", 219, 219);
    HMSetHookSwitch (PORT, ONHOOK);
    asiquit (PORT);
    cprintf ("%c%c", 219, 219);
}

```


SERVER.C

```

    use (comm_wt); /* reset color etc..*/
}

/*-----
open_port:
-----*/
open_port ()
{
    int stat;

    stat = ASSUCCESS;
    if ((stat = asifirst (PORT,MODE,RXLEN,TXLEN)) < ASSUCCESS){
        error (stat);
    }
    if ((stat = asiinit(PORT,BAUD,PARITY,STOP_BITS,WORD_LENGTH))
        < ASSUCCESS ) {
        error (stat);
    }
    if ( (stat = asdtr(PORT,ON)) < ASSUCCESS)
        error (stat);
    if ( (stat = asrts (PORT,ON)) < ASSUCCESS)
        error (stat);
    if ( (stat = asistart(PORT,ASINOUT)) < ASSUCCESS)
        error (stat);
    HMWaitForOK (TICKS_PER_SECOND*3,NULL); /* wait 3 secs */
    HMSetUpAbortKey (ESC);
}

/*-----
init_modem : initialize modem Recursive function
-----*/
int init_modem ()
{
    int stat,i;

    i = 0;
    stat = HMReset (PORT); /* reset modem */
    while ( (stat < ASSUCCESS) && (i <= 3) ){
        ++i;
        stat = HMReset (PORT);
        gotoxy (1,3);
        hang_up ();
        open_port();
    }
    if (i > 3) {
        hang_up ();
        open_port ();
        while (init_modem () < 1) {
            errrtn ("Couldn't Reset Modem. Contact
Central");
        }
        return ( TRUE );
    }

    if (stat < ASSUCCESS)

```


SERVER.C

```

        error (stat);
    if (ECHO == 0)
        if ( (stat = HMSetEchoMode (PORT,OFF))
<ASSUCCESS) /* set echo */
            error (stat);
    if (ECHO == 1)
        if ( (stat = HMSetEchoMode (PORT,ON)) <ASSUCCESS)
            error(stat);
    if ( (stat = HMSetVerboseMode (PORT,ON)) < ASSUCCESS)
        error (stat);
        /* verbal response */
    if ( (stat = HMSetFullDuplexMode (PORT,ON)) < ASSUCCESS)/*
duplex FULL */
        error (stat);
    if ( (stat = HMSetSpeaker (PORT,SPEAKER)) <ASSUCCESS) /*
set speaker */
        error (stat);

    if (i>3)
        return FALSE;

    return TRUE;
}

```


STARTRTB.C

```

/*-----
-
MODULE startrtb.c

PURPOSE: This module does the initialization of the phone for the real-
time
        billing event. Upon a return of a phone, the realtime.c MODULE
performs
        the necessary actions to return and calculate phone charges.
Together
        these two modules are the realtime billing system.

INCLUDES: rtb.h      - all defines are in here
          rtbfunc.h - common functions between startrtb and
realtime occur
                                   here.

Written By : Greg McGregor 1990

REVISED:          What was revised?
GMM 7-30-1991      Nothing
-----
*/

#include <stdio.h>
#include <stdlib.h>
#include <gkeys.h>
#include <bios.h>
#include <time.h>
#include <windows.h>
#include <gbase.h>
#include <extnvar.h>
#include <extscrns.h>
#include <rtb.h>      /* realtime billing definitions */
#include <rtbfunc.h>  /* common rtb functions */
#include <misc.h>
#include <decphone.h>
#include <phonstat.h>

#include <bench.h>
#include <proc.io>
#include <agreev3.h>
#include <phone.h>
#include <agrio.h>

#include <cti_com.h> /* interrupt driven IO to cti */

#define COM1 0
#define COM2 1
#define COM3 2
#define COM4 3

#define CTI_BAUD 9600L
#define CTI_PORT COM1

```


STARTRTB.C

```

/*
 * GLOBALS
 */

cti_obj sco; /* starting CTI object, sco */
char sco_buff[1024]; /* setup an sco buffer of 1K bytes */

/*-----
map_initial maps RTB activity for initial rental
-----*/
int map_initial (s)
int s; /* current state of transmission */
{
    switch (s) {
        case START_STATE:
            return (GET_NUMBER);
        case UNLOCK_PHONE: /* SKIP THIS FOR NOW-UNLOCK OCCURS
AFTER PRINTING*/
            return (GET_NUMBER);
        case GET_VERSION:
            return (GET_NUMBER);
        case GET_NUMBER:
            return (RESET_POINTER);
        case RESET_POINTER:
            return (RESET_CALL_CTR);
        case RESET_CALL_CTR:
            return (RESET_METER);
        case RESET_METER:
            return (SEND_TIME_DATE);
        case SEND_TIME_DATE:
            return (GET_TIME_DATE);
        case GET_TIME_DATE:
            return (POWER_DOWN);
        case POWER_DOWN:
            return (END_STATE);
        case NAK_STATE: /* time didn't set try
again */
            return (SEND_TIME_DATE);
        case END_STATE: /* maps to itself */
            return (END_STATE);
        case ERROR_STATE: /* errors, start over */
            return (START_STATE);
    }
    return (ERROR_STATE); /* STATE_DOESN'T make sense */
}

/*-----
start_rtb
-----*/

```


STARTRTB.C

```

int start_rtb ()
{
    int stat;

    set_up_cti_object (&sco, CTI_BAUD, CTI_PORT, 1024, 2);
    set_cti_buffer (&sco, &sco_buff); /* point sco buffer to sco_buff
*/
    stat = open_cti_port (&sco); /* start interrupts */

    if (stat < 0) {
        rtb_error (-2);
        return (-2);
    }

    stat = start_transfer();

    close_cti_port (&sco);

    if (stat <= 0)
        return (stat);
    return TRUE;
}

```

```

/*-----
start_transfer
-----*/
int start_transfer()
{
    register int out,in,stat;
    int trys;
    int state = START_STATE;
    int DONE = FALSE;
    int pos = 0;
    char abyte;
    int trys_get_time = 0;

    abyte = 0x01;
    while (1) {

        /* determine state of transmission and do appropriate */
        switch (state) {
            case START_STATE:
                trys = 0;
                stat = FALSE;
                set_cti_command (&sco,TURN_ON);
                stat = do_cti_command (&sco);
                if (!stat) {
                    ++trys;
                    errrtn ("Please turn the phone on and place it in the C
TI");
                } else state = map_initial (state);

```


STARTRTB.C

```

        if (!stat) {
            rtb_error (-7);
            return -7;
        }
        break;
case UNLOCK_PHONE:
    set_cti_command (&sco, UNLOCK_PHONE);
    stat = do_cti_command (&sco);
    if (!stat){
        rtb_error (-8);
        return -8;
    } else state = map_initial (state);
    break;
case GET_VERSION:
    set_cti_command (&sco, GET_VERSION);
    stat = do_cti_command (&sco);
    if (!stat){
        rtb_error (-8);
        return -8;
    } else state = map_initial (state);
    break;
case GET_NUMBER:
    use (CTI_wt);
    clrscr ();
    cprintf ("-* Retrieving Cellular Phone Number");
    set_cti_command (&sco, GET_NUMBER); /* get data
into sco_buff */

    stat = do_cti_command (&sco);
    if (!stat) {
        rtb_error (-9);
        return -9;
    } else
        decode_phone (agreemntrec.curphoneno, sco_buff);
    if (!check_phone ()) {
return -23;
    } else state = map_initial (state);
    break;
case RESET_POINTER:
    use (CTI_wt);
    clrscr ();
    cprintf ("-* Resetting Call(s) Pointer");
    set_cti_command (&sco, RESET_POINTER);
    stat = do_cti_command (&sco);
    if (!stat){
        rtb_error (-13);
        return -13;
    } else state = map_initial (state);
    break;
case RESET_CALL_CTR:
    use (CTI_wt);
    clrscr ();
    cprintf ("-* Resetting Call Counter");
    set_cti_command (&sco, RESET_CALL_CTR);
    stat = do_cti_command (&sco);

```


STARTRTB.C

```

        if (!stat){
            rtb_error (-17);
            return -17;
        } else state = map_initial (state);
        break;
case RESET_METER:
    use (CTI_wt);
    clrscr ();
    cprintf ("-* Resetting Cumulative Meter");
    set_cti_command (&sco,RESET_METER);
    stat = do_cti_command (&sco);
    if (!stat){
        rtb_error (-14);
        return -14;
    } else state = map_initial (state);
    break;
case SEND_TIME_DATE:
    use (CTI_wt);
    clrscr ();
    cprintf ("-* Setting RTB Chip Time");
    strcpy (phone_time_rec,"11111111"); /* set to
difference */

    strcpy (phone_time_send,"22222222");
    convert_to_phone_time (phone_time_send);
    set_cti_command (&sco,SEND_TIME_DATE);
    set_cti_send_buffer (&sco,&phone_time_send);
    stat = do_cti_command (&sco);
    set_cti_command (&sco,IN_CTI);
    stat = do_cti_command (&sco);
    if (!stat) {
        rtb_error (-15);
        return -15;
    } else state = map_initial (state);
    break;
case GET_TIME_DATE:
    use (CTI_wt);
    clrscr ();
    cprintf ("-* Verifying RTB Chip Time");
    set_cti_command (&sco,GET_TIME_DATE);
    stat = do_cti_command (&sco);
    moveX (phone_time_rec,sco_buff,10);
    if (!stat){
        rtb_error (-15);
        return -15;
    } else {
        phone_time_rec[0] = 0;
        phone_time_rec[1] = 0;
        phone_time_send[0] = 0;
        phone_time_send[1] = 0;
        /* turns bit 5, of 0-7, off on for some
reason ? */

        phone_time_rec[5] = phone_time_rec[5] &
0xBF; /* bit 5 off */

        if (Xcmp

```


STARTRTB.C

```

(phone_time_send, phone_time_rec, 8) == 0) {
    state = map_initial (state);
} else {
    state = SEND_TIME_DATE;
    clrscr ();
    cprintf ("-* ERROR retrying
time/chip setting!");

    delay (1000); /* delay 1 second */
    ++trys_get_time;
}

if (trys_get_time >= 2) { /* try loop twice */
    rtb_error (-15);
    return -15;
    state = END_STATE;
} else stat = map_initial ( state );
break;
case POWER_DOWN:
    use (CTI_wt);
    clrscr ();
    cprintf ("-* Turning Cellular Phone OFF");
    set_cti_command (&sco, POWER_DOWN);
    stat = do_cti_command (&sco);
    if (!stat){
        rtb_error (-19);
        return -19;
    } else state = map_initial (state);
    break;
case END_STATE:
    stat = end_state_startrtb ();
    if (!stat) {
        rtb_error (-12);
        return -12;
    } else return TRUE;
    break;
case ERROR_STATE:
    delay (1000); /* wait 1000 ms */
    rtb_error (-6);
    return -6;
    state = map_initial (state);
    break;
}

return (TRUE);
}

/*-----
-----
reset_call_counter
-----*/
reset_call_counter ()
{
int stat;

```


STARTRTB.C

```

    use (CTI_wt);
    clrscr ();
    cprintf (" -* Reseting Call Counter!");
    wait_command ();
    stat = bioscom (1, RESET_CALL_CTR, RTB_PORT);
    return TRUE;
}

/*-----
end_state_startrtb
-----*/
int end_state_startrtb ()
{
    use (CTI_wt);
    clrscr ();
    cprintf (" -* Do NOT Remove Phone From CTI!");
    return TRUE;
}

/*-----
check_phone:
-----*/
int check_phone ()
{
    int iostat = 0;
    struct phone_def temprec;
    char s[80];
    iostat = selectinx9 (fd_phone, 1);
    iostat = reset_file9 (fd_phone, &temprec);
    moveX (phonerec.curphoneno, agreemntrec.curphoneno, 12);
    iostat = exactkey9 (fd_phone, &phonerec);
    if (iostat < 0){
        phonerec.curphoneno[12] = '\0'; /* null it */
        display_phone_status_message ('9', phonerec.curphoneno);
        return (FALSE);
    } else
    if (phonerec.status[0] != '0'){
        phonerec.curphoneno[12] = '\0'; /* null it */
        display_phone_status_message
        (phonerec.status[0], phonerec.curphoneno);
        return (FALSE);
    }
    return (TRUE);
}

/*-----
other_phone
-----*/
int other_phone () {
    int correct;
    char ch;

```


STARTRTB:C

```

wintype win;
int key;

win = note ("Put a Different Phone in CTI and Press <ESC> Key");
gotoxy (15,3);
cprintf ("Press the <F2> key to Quit!");
do {
    ch = getch();
    if ( (is_extended_key (ch,&key)) && (key == K_F2) ) {
        windowclose (win);
        return FALSE;
    }
    if (ch == K_ESC) correct = TRUE;
} while (!correct);
windowclose (win);
return TRUE;
}

int unlock_turn_off_phone () {
int stat;

set_up_cti_object (&sco, CTI_BAUD, CTI_PORT, 1024, 2);
set_cti_buffer (&sco, &sco_buff); /* point sco buffer to
sco_buff */

stat = open_cti_port (&sco); /* start interrupts */
use (CTI_wt);
clrscr ();
cprintf ("-* Unlocking Phone");
set_cti_command (&sco,UNLOCK_PHONE);
do_cti_command (&sco);
clrscr ();
cprintf ("-* Turning Phone Off");
set_cti_command (&sco,POWER_DOWN);
do_cti_command (&sco);
/*
set_cti_command (&sco,TURN_OFF);
do_cti_command (&sco);
*/
close_cti_port (&sco);
}

```


TAUSTAT.C

```

177
/*-----
--
update_tau_status

Update function for TAU status byte

GMM 8-10-1991
-----
--*/

#include <stdio.h>
#include <ctype.h>
#include <bench.h>
#include <proc.io>
#include <sys\stat.h>
#include <windows.h>
#include <gkeys.h>
#include <misc.h>

#include <agreev3.h> /* struct formats */
#include <control.h>
#include <phone.h>
#include <raperson.h>
#include <agrio.h>
#include <gbase.h>
#include <time.h>
#include <extnvar.h>
#include <gstring.h>

/*-----
update_tau_status: follow priority values
                    gets reset only if val is of higher priority
-----*/
void update_tau_status (int byte_num,char val) {
    switch (byte_num) {
        case 0:
            agreemntrec.tau_status0[0] = val;
            break;
        case 1:
            agreemntrec.tau_status1[0] = val;
            break;
        case 2:
            agreemntrec.tau_status2[0] = val;
            break;
        case 3:
            agreemntrec.tau_status3[0] = val;
            break;
        case 4:
            agreemntrec.tau_status4[0] = val;
            break;
    }
}

```


UPDAGR.C

```
/*-----
```

```
MODULE: Update Agreement
```

```
Written By      : Greg McGregor 12/1990
```

```
Purpose : To allow an employee to add additional equipment to an already
        completed contract.
```

```
REVISED:          What was revised?
```

```
GMM 7-30-1991      Nothing
```

```
GMM 8-29-1991      changed include <agreemnt.h> to <agreev3.h>
```

```
-----*/
```

```
#include <process.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <window.h>
#include <dos.h>
#include <bios.h>
#include <ctype.h>
#include <bench.h>
#include <proc.io>
#include <\sys\stat.h>
```

```
#include <agriio.h>
#include <agreev3.h> /* all types, making them externs */
#include <control.h>
#include <phone.h>
#include <raperson.h>
#include <gbase.h>
#include <extnvar.h> /* patches global variables as externs */
```

```
#include <windows.h>
#include <gkeys.h>
#include <extscrns.h>
#include <whatopen.h>
#include <misc.h>
#include <getline.h>
#include <cardrdr.h>
#include <credit.h>
#include <dispopen.h>
#include <printer.h>
#include <startrtb.h>
#include <rtbfunc.h>
#include <mainmenu.h>
```

```
windef
```


UPDAGR.C

```
update_win={10,12,70,20,White,Black,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
             White,Black};
```

```
/*-----
```

```
--
```

```
updagr : ENTRY POINT INTO MODULE
```

```
-----
```

```
*/
```

```
updagr ()
```

```
{
```

```
    PRINTED_CONTRACT = FALSE;
```

```
    textcolor (White);
```

```
    textbackground (Black);
```

```
    window (1,1,80,25);
```

```
    clrscr ();
```

```
    mainTitleWindow_update();
```

```
    open_files();
```

```
    if (!get_contract ()) {
```

```
        close_files ();
```

```
        close_all_windows ();
```

```
    } else {
```

```
        update_screen ();
```

```
        close_files ();
```

```
        close_all_windows ();
```

```
    }
```

```
    PRINTED_CONTRACT = FALSE;
```

```
    return;
```

```
}
```

```
/*-----
```

```
update_agreemnt
```

```
-----*/
```

```
update_agreemnt ()
```

```
{
```

```
    int iostat;
```

```
    wintype win;
```

```
    iostat = updrec9 (fd_agreemnt, &agreemntrec);
```

```
    if (iostat < 0){
```

```
        win = note ("Could Not Update The Agreement!");
```

```
        gotoxy (25,3);
```

```
        cprintf ("Press ESC to Exit");
```

```
        getch ();
```

```
        windowclose (win);
```

```
        close_file9 (fd_agreemnt);
```

```
        exit (0);
```

```
    } else {
```

```
        system ("ccopyit agreemnt. ");
```

```
        system ("ccopyit phone. ");
```

```
    }
```

```
}
```


UPDAGR.C

181

```

/*-----
--
mainTitleWindow_update
-----
--*/
mainTitleWindow_update ()
{
windef list_win={2,2,20,9,White,Red,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
                    White,Red};
static wintype list_wt_s;

        main_wt = windowopen (&main_win);
        setttitle (main_wt,"*- Telemac Cellular Corporation
--*",CenterUpperTitle);
        list_wt_s = windowopen (&list_win);
        setttitle (list_wt_s,"COMMANDS",CenterUpperTitle);
        gotoxy (5,2);
        cprintf ("F1 - Help");
        gotoxy (5,3);
        cprintf ("F2 - Cancel");
        gotoxy (5,4);
        cprintf ("F3 - Finish");
        gotoxy (5,5);
        cprintf ("F6 - Exit ");
        use (main_wt);
}

/*-----
-----
get_contract : Pull up a contract to alter
-----
---*/
int get_contract ()
{
wintype win,win2;
pick_list_type menu;
char agreeno_save[15];
int sel,key,iostat,keymatch,found;
struct agreemnt_def temp_agreemnt;

windef
menu_update_win={25,8,55,16,White,Red,FALSE,FALSE,FALSE,TRUE,SINGLEFRAM
E,
                    White,Red};

        found = FALSE;
        add_to_pick_list (&menu,"By Contract Number",1);
        add_to_pick_list (&menu," By Phone Number ",2);
        add_to_pick_list (&menu,"      Exit      ",3);
select:
        found = FALSE;
        null_field (agreemntrec.agreeno,12);
        moveX (agreemntrec.custname,"",25);

```


UPDAGR.C

```

moveX (agreemntrec.curphoneno," - - ",12);
win = windowopen (&menu_update_win);
settitle (win,"Options",CenterUpperTitle);
sel = pick_list (&menu,3,"Select a Method");
switch ( (char)sel ) {
    case 1: windowclose (win);
            get_contract_number ();
            key = 1;
            break;
    case 2: windowclose (win);
            get_phone_number ();
            key = 3;
            break;
    case 3: return ( FALSE ); /* requested Exit */
    case 0x1B: return ( FALSE ); /* ESC key */
}
if (key == 1) {
    iostat = 0;
    iostat = exactkey9 (fd_agreemnt, &agreemntrec);
    if (iostat < 0) {
        errrtn ("Can't Find Agreement!");
        goto select;
    } else found = TRUE;
}
if (key == 3) {
    iostat = 0;
    iostat = reset_file9 (fd_agreemnt,
&temp_agreemnt);
    iostat = exactkey9(fd_agreemnt,
&agreemntrec);
    if (iostat < 0) {
        errrtn ("Can't Find Agreement!");
        goto select;
    } else found = TRUE;
    do{
moveX(agreeno_save,agreemntrec.agreeno,12);
iostat = nextkey9(fd_agreemnt,
&agreemntrec);
if (iostat == 0){
moveX(agreeno_save,agreemntrec.agreeno,12);
}
} while (iostat == 0);
selectinx9(fd_agreemnt, 1); /* read
using agreement number */
moveX(agreemntrec.agreeno,agreeno_save,12);
iostat = exactkey9(fd_agreemnt,
&agreemntrec);
}
if (!found) goto select;
if (agreemntrec.netdue != 0.0) {
    strcpy (errmsg,"This Agreement Has Already
Been Closed!");
}

```


UPDAGR.C

```

errrtn(errmessage);
goto select;
    }
    return found;
}

/*-----
get_contract_number
-----*/
/
get_contract_number ()
{
wintype win;
char number [20];
int key;
    win = windowopen (&manual_win);
    setttitle (win,"Contract Number",CenterUpperTitle);
    strcpy (number,"");
    key = get_line (number,5,1,8,win,"Contract Number -> ");
    selectinx9 (fd_agreemnt,1);
    moveX (agreemntrec.agreeno,number,8);
    windowclose (win);
}

/*-----
get_phone_number
-----*/
get_phone_number ()
{
wintype win;
char number [20];
int key;
    win = windowopen (&manual_win);
    setttitle (win,"Phone Number",CenterUpperTitle);
    strcpy (number,"");
    key = get_line_mask (number,5,1,12,win,"Phone Number -> ",
- - - - -");
    selectinx9 (fd_agreemnt,3);
    moveX (agreemntrec.curphoneno,number,12);
    windowclose (win);
}

/*-----
display_no_batteries_update
-----*/
display_no_batteries_update()
{
    gotoxy (42,3);
    cprintf ("%2.0f",agreemntrec.nobatrent);
}

/*-----

```


UPDAGR.C

```

get_batteries_update
-----*/
int get_batteries_update(wintype win)
{
char s[20];
int stat;

    itoa (agreemntrec.nobatrent,s,10);
    stat = get_line (s,15,3,2,win,"Number of Extra Batteries:
");
    while (!isdigit (s[0])) {
        if (!isdigit (s[0])) {
            strcpy (errmsg,"Must Be Numeric 0 - 9");
            errrtn(errmessage);
        }
        stat = get_line (s,15,3,2,win,"Number of Extra
Batteries: ");
    }
    agreemntrec.nobatrent = atof (s);
    return stat;
}

/*-----*/
display_no_chargers_update
-----*/
void display_no_chargers_update ()
{
    gotoxy (42,5);
    cprintf ("%2.0f",agreemntrec.nochgrent);
}

/*-----*/
get_chargers_update
-----*/
int get_chargers_update (wintype win)
{
char s[20];
int stat;

    itoa (agreemntrec.nochgrent,s,10);
    stat = get_line (s,15,5,2,win,"Number of Chargers
");
    while (!isdigit (s[0])) {
        if (!isdigit (s[0])) {
            strcpy (errmsg,"Must Be Numeric 0 -
9");
            errrtn(errmessage);
        }
        stat = get_line (s,15,5,2,win,"Number of Chargers
: ");
    }
    agreemntrec.nochgrent = atof (s);
    return stat;
}

```


UPDAGR.C

```

/*-----
display_scr1_update()
-----*/
void display_scr1_update()
{
    gotoxy (15,3);
    cprintf ("Number of Extra Batteries:");
    gotoxy (15,5);
    cprintf ("Number of Chargers      :");
}

/*-----
display_values_scr1_update()
-----*/
void display_values_scr1_update()
{
    display_no_batteries_update ();
    display_no_chargers_update ();
}

/*-----
show_agreemnt()
-----*/
show_agreemnt()
{
    static wintype win;
    char s[80];
    windef agreemnt_win
    ={25,5,70,9,White,Red,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
      White,Red};

    win = windowopen (&agreemnt_win);
    setttitle (win,"Contract Information",CenterUpperTitle);
    gotoxy (5,1);
    cprintf ("Agreement #   : ");
    cprintfN (agreemntrec.agreeno,12);
    gotoxy (5,2);
    cprintf ("Customer Name: ");
    cprintfN (agreemntrec.custname,24);
    gotoxy (5,3);
    cprintf ("Portable #   : ");
    cprintfN (agreemntrec.curphoneno,12);
}

/*-----
update_screen

```


UPDAGR.C

```

-----
*/
update_screen (void)
{
    char s[80];
    wintype upd_win, win;
    int FIELD = 1;
    int done, key;

    show_agreemnt();
    done = FALSE;
    upd_win = windowopen (&update_win);
    setttitle (upd_win, "*- Additional Equipment Changes
-*", CenterUpperTitle);
    cursoron ();
    display_scr1_update();
    display_values_scr1_update();
    while (!done) {
        switch (FIELD) {
            case 1: key = get_batteries_update (upd_win);
                    break;
            case 2: key = get_chargers_update (upd_win);
                    break;
        }
        if (UP_FIELD) {
            if (FIELD > 1) {
                --FIELD;
            } else if (FIELD == 1) FIELD = 2;
        }
        if (DOWN_FIELD) {
            if (FIELD < 2) {
                ++FIELD;
            } else if (FIELD == 2) FIELD = 1;
        }

        if (key == FORCED_EXIT) done = TRUE;

        if (key == K_F1) {
            help_list_update ();
        }
        if (key == K_F2) {
            if (PRINTED_CONTRACT) {
                errrtn ("Can't cancel after printing has
been done!");
            } else {
                win = windowopen (&error_win);
                setttitle (win, " F2 - CANCEL!

", CenterUpperTitle);

                gotoxy (5, 2);
                if (yes_no ("Changes will be LOST, Are
you sure (Y/N)?", FALSE)) {
                    centerPrint (60, "Wait A Minute
While I Shut Everything Down!");
                    done = TRUE;

```


UPDAGR.C

```

                                }
                                windowclose (win);
                            }
                        }
if (key == K_F3) {
    update_agreemnt ();
    print_contract (0,FALSE); /* Update contract printing == 0
*/
    if (prt_error_number != 0){
        strcpy (errmsg,prt_error_message);
        errrtn (errmsg);
    } else PRINTED_CONTRACT = TRUE;
}

    if (key == K_F6) {
        if (!PRINTED_CONTRACT) {
            strcpy (errmsg,"You Need To Print The
New Contract!");
            errrtn(errmsg);
        } else {
            update_agreemnt ();
            done = TRUE; /* never should get
here */
        }
    }
}

windowclose (win);
}

/*-----
-----
help_list_update: show command list
-----
--*/
help_list_update ()
{
wintype win;
char c;

    win = windowopen (&commands_win);
    setttitle (win," Quick Step Help ",CenterUpperTitle);
    gotoxy (1,2);
    cprintf ("                STEP");
    gotoxy (1,3);
    cprintf ("                ----");
    gotoxy (1,5);
    cprintf ("    1  -  Change Rental Information");
    gotoxy (1,6);
    cprintf ("    2  -  Press F3 To Finish");
    gotoxy (1,7);
    cprintf ("    3  -  Press F6 To Exit");
    gotoxy (1,8);

```


188

UPDAGR.C

```
cprintf ("      4  -  You're all done!");  
gotoxy (1,10);  
cprintf ("          ESC - EXIT ");  
while ((c = getch ()) != K_ESC) ;  
windowclose (win);
```

```
}
```


189

WHATEND.C

```

/*-----
MODULE : whatend.c  Ending agreement

PURPOSE:  To Monitor what has been done and what still remains
           to be done.  This information will help hold the employees
           hand during a session.

Written By: Greg McGregor 1990

REVISED:          What was revised?
GMM 7-30-1991      Nothing
-----*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <windows.h>
#include <gbase.h>
#include <getline.h>
#include <extnvar.h>
#include <extscrns.h>
#include <gkeys.h>

#define STEPS_END 7

/*
 * GLOBALS
 */
int w_log_table[STEPS_END+1]; /* +1 is overflow space */
int w_init = FALSE;

/*-----
-----
w_log_end : log an event done
-----
---*/
w_log_end (int event)
{
    int i;
    if (!w_init) {
        for (i=0;i<STEPS_END;i++)
            w_log_table[i] = FALSE;
        w_init = TRUE;
    }
    /* now log it done */
    w_log_table[event-1] = TRUE;
}

/*-----
-----
w_init_end : init_log
-----

```


Page 1

WHATEND.C

----*/
w_init_end () {
int i;
for (i=0;i<STEPS_END;i++)
w_log_table[i] = FALSE;
w_init = TRUE;
}

/*-----

w_next_end : what is left to do

----*/
w_next_end (int *FIELD)
{
int i,do_next;
char c;

do_next = -1; /* ERROR */
note_wt = windowopen (¬e_win);
settitle (note_wt,"What To Do Next!",CenterUpperTitle);

WHATOPEN.C

```

/*-----
-----
MODULE: whatopen.c

PURPOSE: Is a module that keeps track of what an employee has done
        versus what is left to do on the initial agreement

Written By : Greg McGregor

REVISED:          What was revised?
GMM 7-30-1991      Nothing
-----*/

#include <stdio.h>
#include <windows.h>
#include <gkeys.h>
#include <misc.h>
#include <time.h>
#include <\h2\hdr\getline.h>
#include <\h2\hdr\gbase.h>
#include <\h2\hdr\extnvar.h> /* vars for W_? type */

#define W_MAX_STEPS 15;      /* how many steps above */

struct w_log_struct {
    int account_log[20]; /* add extra room for no pointer probs */
};

struct w_log_struct w_log_account;

/*-----
init_log_open
-----*/

init_log_open ()
{
    int i;
    for (i=1;i<=15;i++)
        w_log_account.account_log[i] = FALSE;
}

/*-----
-----
w_log : log a step done!
-----*/

void w_log_open (int step)
{
    w_log_account.account_log[step] = TRUE; /* log it as completed
    */
}

```


WHATOPEN.C

```

/*-----
--
w_is_logged_open : is something logged          :PREDICATE
-----
-*/
int w_is_logged_open (int what)
{
    return (w_log_account.account_log[what]);
}

/*-----
-----
what_next_open : start the what next process
-----
---*/
int what_next_open ()
{
    wintype win;
    int i,done;
    char ch;
    done = FALSE;
    i = 1;
    while (!done) {
        if (!w_log_account.account_log[i]){
            switch (i) {
                case 1 :
                    win = note ("Phone Initialization NOT Completed!
PRESS <ESC> <F4>");
                    done = TRUE;
                    break;
                case 2 :
                    win = note ("Credit Authorization NOT Obtained!
PRESS <ESC> <F3>");
                    done = TRUE;
                    break;
                case 3 :
                    win = note ("Enter Customer Name!");
                    done = TRUE;
                    break;
                case 4 :
                    win = note ("Enter Customer's Credit Card Number
!");
                    done = TRUE;
                    break;
                case 5 :
                    win = note ("Enter Credit Card Expiration Date!");
                    done = TRUE;
                    break;
                case 6 :

```


WHATOPEN.C

```
win = note ("Enter Customer's Driver's License
Number!");

done = TRUE;
break;
case 7 :
win = note ("Enter Customer's Home Address!");
done = TRUE;
break;
case 8 :
win = note ("Enter Customer's Home City!");
done = TRUE;
break;
case 9 :
win = note ("Enter Customer's Home State!");
done = TRUE;
break;
case 10:
win = note ("Enter Customer's Home Zip Code!");
done = TRUE;
break;
case 11:
win = note ("Enter Customer's Home Phone!");
done = TRUE;
break;
case 12:
win = note ("Must enter
customer's local phone number!");
done = TRUE;
break;
case 13:
win = note ("Must enter
expected rental return date!");
done = TRUE;
break;
case 14:
win = note ("Must tell customer about LDW!");
done = TRUE;
break;
case 15:
win = note ("Press <F5> key to Print Receipt!");
done = TRUE;
break;
default :
win = note ("Press <F6> to Exit. You are all
DONE!");
done = TRUE;
break;
} /* end switch */
gotoxy (1,3);
centerPrint (60,"Press <ESC> to Clear message");
do {
    ch = getch ();
} while (ch != K_ESC) ;
windowclose (win);
```


WHATOPEN.C

```
        } /* end if */  
    ++i;  
    if (i > 16) done = TRUE;  
} /* ends while loop */  
}
```


CTI_COM.C

```
/*-----
MODULE:  CTI_COM
```

This module contains the core functions for reading and writing to the CTI in interrupt mode.

Written By: Greg McGregor

Revisions:

```
-----*/
```

```
/*
 * include files
 */
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <dos.h>
#include <bios.h>
#include <conio.h>
```

```
#include <rtb.h>
#include <cti_com.h>
#include <windows.h>
#include <misc.h>
```

```
/*
 * Greenleaf includes
 */
```

```
#include "gf.h"
#include "asiports.h"
#include "ibmkeys.h"
```

```
#define FALSE 0
#define TRUE 1
```

```
/*-----
 *
 * Procedure Name: send_char_cti
 * Parameters: cti_obj, char ch
 * Function: send a char via asiputc wait on transmit buffer
 * Returns: TRUE, FALSE
 *
 * Written By: Greg McGregor
```


CTI_COM.C

177

```
-----
-*/
int send_char_cti (cti_obj *cd, char ch) {
int stat;
    stat = asiputc (cd->port,ch);
    if (stat < 0) return FALSE;
    stat = wait_transmit_buffer (cd);
    return stat;
}
```

```
/*-----
*
* Procedure Name: get_char_cti
* Parameters: cti_obj
* Function: get char from serial port, CTI on timeout basis
* Returns: the CHAR gotten <OR> -1 for failure
*
* Written By: Greg McGregor
-----*/
```

```
-*/
char get_char_cti (cti_obj *cd) {
int stat;
int ticks = 0;
    while (ticks < cd->time_out*TICKS_PER_SECOND) {
        stat = asigetc (cd->port);
        if (stat >= 0) return ((char)stat);
        timer (1); /* 54.9 ms */
        ++ticks;
    }
    return -1;
}
```

```
/*-----
*
* Procedure Name: set_up_cti_object
* Parameters:
* Function: set port, baud, buffer, time out for communication
*
* Written By: Greg McGregor
-----*/
```

```
*/
void set_up_cti_object (cti_obj *cd, long int baud, int port, int
buff_size, int time_out) {
    cd->baud = baud;
    cd->port = port;
    cd->buff_size = buff_size;
    cd->time_out = time_out;
    return;
}
```


CTI_COM.C

```
/*-----  
*  
* Procedure Name: set_cti_timeout  
* Parameters:  
* Function:  
*  
* Written By: Greg McGregor  
*-----
```

```
*/  
void set_cti_timeout (cti_obj *cd, int time_out) {  
    cd->time_out = time_out;  
    return;  
}
```

```
/*-----  
*  
* Procedure Name: set_cti_send_count  
* Parameters:  
* Function:  
*  
* Written By: Greg McGregor  
*-----
```

```
*/  
void set_cti_send_count (cti_obj *cd, int count) {  
    cd->send_count = count;  
    return;  
}
```

```
/*-----  
*  
* Procedure Name: set_cti_rec_count  
* Parameters:  
* Function:  
*  
* Written By: Greg McGregor  
*-----
```

```
*/  
void set_cti_rec_count (cti_obj *cd, int count) {  
    cd->rec_count = count;  
    return;  
}
```

```
/*-----  
*  
* Procedure Name: set_cti_command  
* Parameters:  
* Function:  
*  
* Written By: Greg McGregor  
*-----
```

```
*/  
void set_cti_command (cti_obj *cd, int command) {
```


CTI_COM.C

```

        cd->command = command;
        return;
    }

```

```

/*-----
 *
 * Procedure Name: set_cti_buffer
 * Parameters:
 * Function:
 *
 * Written By: Greg McGregor
 *-----
 */

```

```

void set_cti_buffer (cti_obj *cd, char *buff) {
    cd->buffer = buff;
    return;
}

```

```

/*-----
 *
 * Procedure Name:
 * Parameters:
 * Function:
 *
 * Written By: Greg McGregor
 *-----
 */

```

```

/*
 *
 * set_cti_send_buffer
 */
void set_cti_send_buffer (cti_obj *cd, char *buff) {
    cd->command_bytes = buff;
    return;
}

```

```

/*-----
 *
 * Procedure Name: open_cti_port
 * Parameters:
 * Function:
 * Returns:
 *
 * Written By: Greg McGregor
 * Revised By: Ted Watler
 * Comments By Greg McGregor:
 *     In talking with Greeleaf software, you should not use
 *     asiflow and asicheck together; however, it hasn't failed
 *     so what aint broke I'm not fixin.
 *-----

```

```

-*/
int open_cti_port (cti_obj *cd) {

```


CTI_COM.C

```
int stat;
```

```
    if ( (stat = asiopen
(cd->port,ASINOUT|BINARY|NORMALRX,cd->buff_size,cd->buff_size,cd->baud,
P_NON
E,1,8,ON,ON)) == ASSUCCESS )
        if ( (stat = asiflow(cd->port, 1, 99, ON, ON)) !=
ASSUCCESS )
            stat = asiquit (cd->port);
        else {
            asicheck(cd->port, CTS_LOW_STOPS_TX_INTERRUPTS, ON);
            asiclear(cd->port,ASINOUT);
        }
```

```
    return(stat);
}
```

```
/*-----
*
* Procedure Name: close_cti_port
* Parameters:
* Function:
* Returns: < 0 failure
*
* Written By: Greg McGregor
-----
-*/
```

```
int close_cti_port (cti_obj *cd) {
    return asiquit (cd->port);
}
```

```
/*-----
*
* Procedure Name: do_cti_command
* Parameters:
* Function:
* Returns:
*
* Written By: Greg McGregor
-----
-*/
```

```
int do_cti_command (cti_obj *command) {
int stat;
int cmd;
int trys = 1;    /* give each command 2 trys to succeed */

while (trys <= 2) {
    stat = do_cti_func (command);
    /* don't do check for command success after these commands */
}
```


CTI_COM.C

```

    if (command->command == POWER_DOWN) return ( stat ) ;

    pause_execution (2);      /* wait x tenths of a second */
    cmd = command->command;
    set_cti_command (command, IN_CTI);
    stat = do_cti_func (command);
    set_cti_command (command, cmd);
    asiclear (command->port, ASINOUT); /* clear ports after
successful command */
    ++trys;
        if (stat) return TRUE;

            /* stat is false here */
    set_cti_command (command, TURN_ON); /* if failed need to turn o
n*/
        do_cti_func (command);
        set_cti_command (command, cmd);
    }
    if (stat == FALSE) {      /* attempt to turn cti off */
        set_cti_command (command, POWER_DOWN);
        do_cti_func (command);
    }
    return FALSE;
}

/*-----
*
* Procedure Name: pause_execution
* Parameters: tenths seconds
* Function: ANSI comp
* Returns: NONE
*
* Written By: Greg McGregor
*-----
*/
void pause_execution (int tenths_secs)
{
    clock_t start, current;
    start = clock ();
    current = clock ();
    while ( (((int)(current - start)*10) / CLK_TCK) < tenths_secs)
        current = clock ();
    return;
}

/*-----
*
* Procedure Name: wait_transmit_buffer
* Parameters:
* Function: HARDWARE DEPENDENT TIMEOUT!
* Returns: TRUE, FALSE (True if buffer emptied )
*

```


CTI_COM.C

* Written By: Greg McGregor

```

-*/
int wait_transmit_buffer (cti_obj *cd) {
int stat = TRUE;
long int i = 0;
clock_t start,end;

    start = clock ();
    while (stat) {
        if (istxempty (cd->port)) {
            stat = FALSE;
        } else stat = TRUE;
        ++i;
        /* can't use clock other end times out at
3 ms */
        if ( (!istxinrunning (cd->port)) && (iscts (cd->port,IMMEDIATE
)) ){
            asiresume (cd->port,ASINOUT);
        }
        end = clock ();
        if ( ((end - start)/CLK_TCK) >= cd->time_out) {
            return FALSE;
        }
    }
    return TRUE;
}

```

```

/*
*
* Procedure Name: send_cti_data
* Parameters:
* Function: send data in command_bytes field
* Returns: TRUE, FALSE
*
* Written By: Greg McGregor

```

```

-*/
int send_cti_data (cti_obj *cd) {
int i;
    for (i=0;i<cd->send_count;i++) {
        if (!send_char_cti (cd,cd->command_bytes[i]))
            return FALSE;
    }
    return (i);
}

```

```

/*
*
* Procedure Name: get_cti_data_timed
* Parameters:
* Function: get X bytes based on time out

```


CTI_COM.C

* Returns: FALSE <OR> number of bytes gotten

*

* Written By: Greg McGregor

*/

```
int get_cti_data_timed (cti_obj *cd) {
int i;
int stat = 0;
int count = 0;
    i = 0;
    if (cd->rec_count == -1) { /* get as many bytes as possible */
        while (stat >= 0) {
            stat = (int)get_char_cti (cd);
            if (stat == -1) return FALSE;
            cd->buffer[i] = (char)stat;
                ++i;
            cd->rec_count = i;
        }
        cd->buffer[i] = '\0';
    } else {
        while (i < cd->rec_count) {
            ++count;
            if (count >= 9) {
                clrscr ();
                cprintf ("-* Got %d of %d Data
Bytes", i, cd->rec_count);
                count = 0;
            }
            stat = (int)get_char_cti (cd);
            if (stat == -1) return FALSE;
            cd->buffer[i] = (char)stat;
                ++i;
        }
        cd->buffer[i] = '\0'; /* end string with a null */
    }
    clrscr ();
    cprintf ("-* Got %d of %d Data Bytes", i, cd->rec_count);
    cd->rec_count_got = i; /* set number of bytes we got */
    return (i); /* the number of bytes we got */
}
```

/*-----

*

* Procedure Name: do_cti_func

* Parameters:

* Function: do the cti command

* Returns: TRUE, FALSE on success or failure

*

* Written By: Greg McGregor

*/

```
int do_cti_func (cti_obj *cd) {
int stat;
```


CTI_COM.C

```

int ch;
int stat_ch;
switch (cd->command) {
case SEND_TIME_DATE:
    cd->send_count = 8;
    if (!send_char_cti (cd,SEND_TIME_DATE)) return FALSE;
    if ( (stat = send_cti_data (cd)) < 0) return FALSE;
    return TRUE;
case TURN_OFF:
    if (!send_char_cti (cd,TURN_OFF)) return FALSE;
    timer (44); /* do a 3 second even though doesn't
matter*/
    return TRUE;
case POWER_DOWN:
    if (!send_char_cti (cd,POWER_DOWN)) return FALSE;
    timer (44);
    return TRUE;
case TURN_ON:
    if (!send_char_cti (cd,TURN_ON)) return FALSE;
    timer (44); /* wait 3 seconds before return */
                /* to ensure phone is up and running */
    return TRUE;
case LOCK_PHONE:
    if (!send_char_cti (cd,LOCK_PHONE)) return FALSE;
    timer (44); /* wait 3 second for reboot of phone */
    return TRUE;
case UNLOCK_PHONE:
    if (!send_char_cti (cd,UNLOCK_PHONE)) return FALSE;
    timer (44); /* wait 3 second for reboot of phone */
    return TRUE;
case RESET_POINTER:
    if (!send_char_cti (cd,RESET_POINTER)) return FALSE;
    return TRUE;
case RESET_CALL_CTR:
    if (!send_char_cti (cd,RESET_CALL_CTR)) return FALSE;
    return TRUE;
case RESET_METER:
    if (!send_char_cti (cd,RESET_METER)) return FALSE;
    return TRUE;
case GET_NUMBER:
    ch = GET_NUMBER;
    cd->rec_count = 6; /* 6 bytes for phone number
*/
    break;
case GET_INFO:
    ch = GET_INFO; /* rec_count set externally in
realtime.c */
    /* CTI firmware comp. 8-23-1991 */
    if (!send_char_cti (cd,ch)) return FALSE;
    if (!send_char_cti (cd,(char)(cd->rec_count &
0x00FF))) return FALSE;
    if (!send_char_cti (cd,(char)((cd->rec_count>>8) &
(0x00FF)))) return FALSE;
    stat = get_cti_data_timed (cd);

```


CTI_COM.C

```

        if (cd->rec_count_got != cd->rec_count) return FALSE;
        return TRUE;
    case GET_TIME_DATE:
        ch = GET_TIME_DATE;
        cd->rec_count = 8;        /* 8 bytes for time and
date data */
        break;
    case GET_SERIAL:
        ch = GET_SERIAL;
        cd->rec_count = -1;      /* -1 get as many as you
can */
        break;
    case GET_VERSION:
        ch = GET_VERSION;
        cd->rec_count = 12;
        break;
    case GET_NOVA_VER:
        ch = GET_NOVA_VER;
        cd->rec_count = -1;
        break;
    case GET_MBC_VER:
        ch = GET_MBC_VER;
        cd->rec_count = -1;
        break;
    case READ_METER:
        ch = READ_METER;
        cd->rec_count = 8;
        break;
    case NUMBER_CALLS:
        ch = NUMBER_CALLS;
        cd->rec_count = 2;
        break;
    case GET_POINTER:
        ch = GET_POINTER;
        cd->rec_count = 2;
        break;
    case IN_CTI:
        /* status bits send back */
        /* bit0 = IN CTI */
        /* bit 1 = TRANSFER TIME OUT */
        /* bit2 = COMMAND FAILED TIME OUT */
        if (!send_char_cti (cd, IN_CTI)) return FALSE;
        stat_ch = get_char_cti (cd);
        if (stat_ch == 2) { cd->error_code=2; /* bit 1 on
*/
        } else
        if (stat_ch == 4) { cd->error_code=3; /* bit 2 on
*/
        } else
        if (stat_ch == 0) { cd->error_code=0; /* phone
not in cti*/
        } else
        if (stat_ch == -1) { cd->error_code=4; /* timed
out on getc */

```


CTI_COM.C

```
        } else if (stat_ch & 0x01)    cd->error_code = 1;
/* no error */

        if (cd->error_code == 1) return TRUE;
        return FALSE;
    }
    if (!send_char_cti (cd,ch)) return FALSE;
    stat = get_cti_data_timed (cd);
    if (stat > 0) return TRUE;
    return FALSE;
}
```


DATES.C

```

/*-----
MODULE: Dates.c
Written by PRO-C

REVISED:          What was revised?
- GMM 7-30-1991    Modified it back in 1990
-----*/

int day_tab[][13] =
{
    {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
    {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}
};

/****( dates.c )****
**/
/*
*/
/* PRO-C - Copyright (c) 1988 Vestronix Inc.
*/
/* 18 OCT 88
*/
/*
*/
/*****
**/
/*
* PRS
* Tuesday the 11th of October 1988, about a quarter past four in the
afternoon.
* Date io routines - get_date fmt_date and is_date
* see end for documentation.
*/

# include <stdio.h>
# include <ctype.h>
# include <time.h>
# include <bench.h>

/* Function prototypes */
# ifdef ANSI
static int alpha_month(char *);
static int daym(int ,char *);
static int dtype(int ,char *);
static int get_day(void);
static int get_month(void);
static int get_year(void);
static int hour(int ,char *);
static int minute(int ,char *);
static int monname(int ,char *);
static int wday(int ,char *);
static int ynum(int ,char *);

```


DATES:C

```

static void stradd(char *,char *);
static int week_day(int, int, int);
# else
static int alpha_month();
static int daym();
static int dtype();
static int get_day();
static int get_month();
static int get_year();
static int hour();
static int minute();
static int monname();
static int wday();
static int ynum();
static void stradd();
static int week_day();
# endif

```

```

static char buf[81]; /* Returned, so overwritten everytime */
#define DATELEN 6

```

```

char *bptr; /* Pointer into the date buffer */

```

```

/* dstr must be YYYYMMDD */

```

```

char *fmt_date(dstr, mask)
char *dstr, *mask;
{

```

```

    struct tm tim, *tptr, *localtime();
    int i;
    char specials[8];

```

```

    strcpy(specials, "DTMYW"); /* Format characters */
    bptr = buf;

```

```

    if ( dstr == NULL || *dstr == '\0' ) {
        /* get today's date */
        long secs;

```

```

        secs = time((long *)0);
        tptr = localtime(&secs);
        strcat(specials, "HN"); /* allow weekdays and time */
        tptr->tm_mon++;

```

```

    } else {

```

```

        /*
         * Check the date, but not much.
         * Allow for non_NULL terminated string (dstr),
         * but it MUST have space for DATELEN (6) chars
         */
        *buf = '\0';
        for (i = 0; i < DATELEN; i++) {
            if (!isdigit(dstr[i])) {
                if (dstr[i] == ' ')
                    dstr[i] = ' ';
            }

```


DATES.C

```

        else {
            buf[i] = '\0';
            return(buf);
        }
        buf[i] = dstr[i];
    }
    buf[DATELEN] = '\0';
    tptr = &tim;
    /*
    tptr->tm_mday = atoi(&buf[4]);
    buf[4] = '\0';
    tptr->tm_mon = atoi(&buf[2]);
    buf[2] = '\0';
    tptr->tm_year = atoi(buf);
    */
    sscanf(buf, "%2d%2d%2d", &tptr->tm_year, &tptr->tm_mon,
&tptr->tm_mday);
    /* If its complete garbage then return the input string */
    if (tptr->tm_mday < 1 || tptr->tm_mday > 31 ||
tptr->tm_mon < 1 ||
        tptr->tm_mon > 12 || tptr->tm_year < 0 || tptr->tm_year
> 99)
        return(buf);
    tptr->tm_wday = 0;
}

*buf = '\0';
while (*mask) {
    if ((*mask + 1) == *mask) && strchr(specials, *mask)) {
        switch (*mask) {
            case 'W' :
                if (tptr->tm_wday == 0)
                    tptr->tm_wday =
week_day(tptr->tm_year,
tptr->tm_mon, tptr->tm_mday);
                mask += wday(tptr->tm_wday, mask);
                break;
            case 'H' :
                mask += hour(tptr->tm_hour, mask);
                break;
            case 'N' :
                mask += minute(tptr->tm_min, mask);
                break;
            case 'D' :
                mask += daym(tptr->tm_mday, mask);
                break;
            case 'T' :
                mask += dtype(tptr->tm_mday, mask);
                break;
            case 'M' :
                mask += monname(tptr->tm_mon, mask);
                break;
            case 'Y' :

```


DATES.C

```

mask += ynum(tptr->tm_year, mask);
break;
    }
    } else {
        *(bptr++) = *(mask++);
    }
}
return(buf);
}

/* Put these into bench.c and just use fields of 3 for Tue etc, GEO */
char    *lday_name[10] = {
    "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",
    "Friday", "Saturday", "???" };

char    *day_name[] = {
    "Sun", "Mon", "Tue", "Wed", "Thu",
    "Fri", "Sat", "???" };

static int wday(day, mask)
int      day;
char     *mask;
{
    int      nWs;

    for (nWs = 0; *mask == 'W'; mask++)
        nWs++;
    switch (nWs) {
    case 0 :
        return(nWs); /* gash input */
    case 1 :
        *bptr++ = 'W';
        *bptr = '\0';
        return(nWs); /* gash input */
    case 2 :
        *bptr++ = (day / 10) + '0';
        *bptr++ = (day % 10) + '0';
        *bptr = '\0';
        return(nWs);
    case 3 :
    case 4 :
    case 5 :
    case 6 :
    case 7 :
    case 8 : /* treat all like 3, ignore trailing W's */
        stradd(bptr, lday_name[day]);
        bptr += 3;
        return(nWs);
    default : /* >= 9 */
        stradd(bptr, lday_name[day]);
        bptr += strlen(lday_name[day]);
    }
}

```


DATES.C

```

        return(nWs);
    }
}

static int daym(mday, mask)
int      mday;
char     *mask;
{
    int      nDs;

    for (nDs = 0; *mask == 'D'; mask++)
        nDs++;
    switch (nDs) {
    case 0 :
    case 1 :
        return(0); /* gash input */
    default :
        *bptr++ = (mday > 9) ? (mday / 10) + '0' : '0'; /*JH -
leading 0 if day is single digit.*/
        *bptr++ = (mday % 10) + '0';
        *bptr = '\0';
        return(2);
    }
}

static int hour(hr, mask)
int      hr;
char     *mask;
{
    int      nHs;

    for (nHs = 0; *mask == 'H'; mask++)
        nHs++;
    switch (nHs) {
    case 0 :
    case 1 :
        return(0); /* gash input */
    default :
        *bptr++ = (hr / 10) + '0';
        *bptr++ = (hr % 10) + '0';
        *bptr = '\0';
        return(2);
    }
}

static int minute(mn, mask)
int      mn;
char     *mask;
{
    int      nNs;

```


DATES.C

```

for (nNs = 0; *mask == 'N'; mask++)
    nNs++;
switch (nNs) {
case 0 :
case 1 :
    return(0); /* gash input */
default :
    *bptr++ = (mn / 10) + '0';
    *bptr++ = (mn % 10) + '0';
    *bptr = '\0';
    return(2);
}
}

/* Put these into bench.c and just use fields of 3 for Jan etc, GEO */
char smnth[][4] = {
    "???", "Jan", "Feb", "Mar", "Apr", "May",
    "Jun", "Jul", "Aug", "Sep", "Oct",
    "Nov", "Dec",
    "???" };

char lmnth[][10] = {
    "January", "January", "February", "March", "April", "May",
    "June", "July", "August", "September", "October",
    "November", "December",
    "???????" };

static int monname(mnum, mask)
int mnum;
char *mask;
{
    int nMs;

    for (nMs = 0; *mask == 'M'; mask++)
        nMs++;
    switch (nMs) {
case 0 :
    return(0);
case 1 :
    /* crap input */
    *bptr++ = 'M';
    return(1);
case 2 :
    *bptr++ = (mnum / 10) + '0';
    *bptr++ = (mnum % 10) + '0';
    *bptr = '\0';
    return(2); /* gash input */
case 3 :
case 4 :
case 5 :
case 6 :

```


DATES.C

```

case 7 :
case 8 : /* treat all like 3, ignore trailing W's */
        stradd(bptr, smnth[mnum]);
        bptr += 3;
        return(nMs);
default : /* >= 9 */
        stradd(bptr, lmnth[mnum]);
        bptr += strlen(lmnth[mnum]);
        return(nMs);
}
}

```

```
/* Interesting all this */
```

```

char ntype[][3] = {
    "??",
    "st", "nd", "rd", "th", "th", "th", "th", "th", "th", "th",
    "th", "th", "th", "th", "th", "th", "th", "th", "th", "th",
    "st", "nd", "rd", "th", "th", "th", "th", "th", "th", "th",
    "st" };

```

```
static int dtype(mday, mask)
```

```

int mday;
char *mask;
{
    int nTs;

    for (nTs = 0; *mask == 'T'; mask++)
        nTs++;
    switch (nTs) {
    case 0 :
    case 1 :
        return(0); /* gash input */
    default :
        stradd(bptr, ntype[mday]);
        bptr += strlen(ntype[mday]);
        return(2);
    }
}

```

```
static int ynum(yno, mask)
```

```

int yno;
char *mask;
{
    int nYs;

    for (nYs = 0; *mask == 'Y'; mask++)
        nYs++;
    switch (nYs) {
    case 0 :
    case 1 :
        return(0); /* gash input */
    }
}

```


DATES.C

```

case 7 :
case 8 : /* treat all like 3, ignore trailing W's */
        stradd(bptr, smnth[mnum]);
        bptr += 3;
        return(nMs);
default : /* >= 9 */
        stradd(bptr, lmnth[mnum]);
        bptr += strlen(lmnth[mnum]);
        return(nMs);
}

```

```

/* Interesting all this */
char ntype[][3] = {
    "??",
    "st", "nd", "rd", "th", "th", "th", "th", "th", "th", "th", "th",
    "th", "th", "th", "th", "th", "th", "th", "th", "th", "th",
    "st", "nd", "rd", "th", "th", "th", "th", "th", "th", "th",
    "st" };

```

```

static int dtype(mday, mask)
int mday;
char *mask;
{
    int nTs;

    for (nTs = 0; *mask == 'T'; mask++)
        nTs++;
    switch (nTs) {
    case 0 :
    case 1 :
        return(0); /* gash input */
    default :
        stradd(bptr, ntype[mday]);
        bptr += strlen(ntype[mday]);
        return(2);
    }
}

```

```

static int ynum(yno, mask)
int yno;
char *mask;
{
    int nYs;

    for (nYs = 0; *mask == 'Y'; mask++)
        nYs++;
}

```


DATES.C

```

case 2 :
case 3 :
    *bptr++ = (yno / 10) + '0';
    *bptr++ = (yno % 10) + '0';
    *bptr = '\0';
    return(nYs);
default :
    *bptr++ = '1';
    *bptr++ = '9';
    *bptr++ = (yno / 10) + '0';
    *bptr++ = (yno % 10) + '0';
    *bptr = '\0';
    return(nYs);
}

```

/* stradd is similar to strcat, but assumes that s1 points to the END of
a string, ie the \0 ; */

```

static void stradd(s1, s2)
char    *s1, *s2;
{
    while (*s1++ = *s2++)
        ;
}

```

```

char    *get_date(dstr, dtype)
char    *dstr;
int     dtype;
/*

```

* Where dstr is the (null-terminated) date_string to be converted, and
dtype
* determines the type of string it purports to be, see above definitions.
*/

```

{
    int     yr, mnth, day;
    char    retstr[80];

    if (dstr == NULL || *dstr == '\0') { /* null, get today's */
        dtype = YMD;
        strcpy(dstr, fmt_date(NULL, "YYMMDD"));
    }
    bptr = dstr;
    switch (dtype) {
    case YMD :
        yr = get_year(); /* only 2 digits allowed if leading */
                        /* JH - 4 digits if leading */
        mnth = get_month();

```


DATES.C

```

        day = get_day();
        break;
case MDY      :
        mnth = get_month();
        day = get_day();
        yr = get_year();
        break;
case DMY      :
        day = get_day();
        mnth = get_month();
        yr = get_year();
        break;
default :
        return(dstr);
}
if ( yr <= 0 ) { /* get today's date */
        struct tm *tptr, *localtime();
        long      secs;

        secs = time((long *)0);
        tptr = localtime(&secs);
        yr = tptr->tm_year;
}
if (day < 1 || day > 31 || mnth < 1 || mnth > 12 || yr < 0 || yr >
99) {
        return(dstr);
}
sprintf(retstr, "%02d%02d%02d", yr, mnth, day);
return(retstr);
}

```

```

static int get_day()
{
        int      day;

        while (*bptr && !isdigit(*bptr))
                bptr++;
        if (!*bptr)
                return(0);
        day = *bptr - '0';
        bptr++;
        if (isdigit(*bptr)) {
                day *= 10;
                day += *bptr - '0';
                bptr++;
        }
        return(day);
}

```

```

static int get_year()
{
        /* This is only for trailing years and allows four digits */

```


DATES.C

```

long    year;

while (*bptr && !isdigit(*bptr))
    bptr++;
if (!*bptr)
    return(-1);
year = 0;
while (isdigit(*bptr)) {
    year *= 10;
    year += *bptr - '0';
    bptr++;
    if (year > 9999)
        return(-1);
}
if (year > 99)
    year = year % 100;
return((int)year);
}

/*
 * Dig a month out of a string, fast for easy dates (1 1 88), effective
 * for stupid dates ( 1st of January 1988 )
 * Return integer month, hopefully 1..12 but that is not checked here,
 * see is_date for validation (very fancy).
 */
static int get_month()
{
    int    month = 0;

    for (;;) {
        if (!*bptr) /* at end of bptr */
            return(month);
        while (*bptr && !isalnum(*bptr)) /* Find letter or digit */
            bptr++;
        while (isalpha(*bptr)) { /* Try and find an alpha month */
            if (month = alpha_month(bptr))
                return(month); /* found jan or something */
            bptr++;
        }
        if (isdigit(*bptr)) { /* look for a digit month */
            month = *bptr - '0';
            bptr++;
            if (isdigit(*bptr)) {
                month *= 10;
                month += *bptr - '0';
                bptr++;
            }
            return(month); /* Found some digit or other, so
assume that's it */
        }
    }
    /* NOTREACHED */
    return(month);
}

```


DATES.C

```

}

char    mnth[][4] = {
    "jan", "feb", "mar", "apr", "may",
    "jun", "jul", "aug", "sep", "oct",
    "nov", "dec", "000" };

static int alpha_month(mstr)
char    *mstr;
{
    int    c1;

    for (c1 = 0; c1 < 3; c1++) {
        if (*(mstr + c1) == '\0')
            return(0);
        if (isupper(*(mstr + c1)))
            *(mstr + c1) = tolower(*(mstr + c1));
    }
    for (c1 = 0; c1 < 12; c1++)
        if ((*mstr == mnth[c1][0]) &&
            (*(mstr + 1) == mnth[c1][1]) &&
            (*(mstr + 2) == mnth[c1][2]))
            return(c1 + 1);
    return(0);
}

/*
 * Date must be since 1970.
 * There's a routine called Zeller's congruence (.or something ) which
would
 * be much better, but I've lost it.
 */
static int week_day(year, month, day)
int year, month, day;
{
    static int Juldays[] = {0, 0, 31, 59, 90, 120, 151, 181, 212, 243,
                             273, 304, 334};
    /* Don't actually need these checks for this set of functions */
    if (year > 99)
        year -= 1900;
    if ((year -= 70) < 0)
        return(7); /* Don't know */

    return ((int)((long)year * 365L + (long)(Juldays[month] + day + 3 +
        ((year + 1) / 4) + (((year % 4) == 2) && (month > 2)))) %
7);
}

/*****
**/
/* Procedure Name : ISDATE

```


DATES.C

```
*/
/*-----
-*/
/* Check to see if 'in' is in the format DDMMYY.
*/
/*****
**/

int    day_tab[][13];

int    is_date(in)
char   *in;
{
    int    i;
    int    day, month, year;

    for (i = 0; i < 6; i++)
        if (!isdigit(in[i]))
            return(FALSE);

    year = ((in[0] - '0') * 10) + (in[1] - '0');
    month = ((in[2] - '0') * 10) + (in[3] - '0');
    day = ((in[4] - '0') * 10) + (in[5] - '0');

    /* The leap year bit below is good up to 2100. By which time I'll
be 139. */
    if ( year > 99 || year < 0 || month < 1 || month > 12 || day < 1
|| day > day_tab[(year % 4)][month])
        return(FALSE);
    return(TRUE);
}

/*
#if 0
```

PRS Tuesday the 11th of October 1988, five to two in the afternoon.

This attempts to describe how the date format routines are intended to work. There are a couple of known infelicitudes (bugs) which will be fixed by friday.

There are four aspects to the date formatters.

1. How the date is stored internally.
2. How the date is entered by the user
3. How the date is checked.
4. How the date is displayed.

1. All generated programs currently store dates as six characters, in the format YYMMDD. This allows easy comparison / sorting.

2. There are three ways the designer of a generated program may choose to have dates entered :

DATES.C

Day-Month-Year (DMY)
Month-Day-Year (MDY)
Year-Month-Day (YMD)

The default is YMD. There should be a way to change this default, but there isn't. To set the format, choose validation code 5.

The function get_date() accepts user input and attempts to convert it to the format YYMMDD. It does minimal validation, only enough to stop itself from crashing.

It understands a wide range of date formats, so that if date format DMY has been selected then

1-1-88
1/1/88
1 1 1988
The 1st jan 1988.
1st Jannwery year of our lord 1988 (sic)

are all converted to 880101. Note that with YMD input, the year can only be entered as two digits, not four. Anyone who cares to consider the algorithnm necessary to convert 19880101 will understand why.

3. The date is validated by the function is_date() which takes a string in YYMMDD format. The method it uses for checking feb 29 works up to 2100.

4. The date is displayed by the function fmt_date which takes a YYMMDD date and a format string (mask). The mask may contain arbitrary text, which is displayed unchanged, and 'special' formatting characters which will be converted.

PRO-C only allows entry of a 20 character mask, fmt_date accepts a mask of up to 80 characters. The date need not be null-terminated

the mask must. If the date is null then today's date is used.

Special characters are : (NB : Read MMMMMMMMMM for M[+9] in all cases)

D The first DD found will be replaced with the day number, eg 15.

T TT following (perhaps after spaces) DD will be replaced with th, st, nd, rd as appropriate.

M Thee first M[+9] found will be replaced with the long month name.

DATES.C

The first MMM found will be replaced by the short month name, eg Jan.

The first MM found will be replaced by the month number, eg 01.

The first YY found will be replaced by the two-digit year, eg 88.
The first YYYY found will be replaced by the four-digit year, eg 1988.

The following characters only work for today's date, ie a null date.

The first W[+9] found will be replaced by the long weekday name, eg

Monday.

The first WWW found will be replaced by the short weekday name, eg Monday.

The first WW found will be replaced by the weekday number, Sun = 0

The first HH found will be replaced by the two-digit hour, eg 11.

The first NN found will be replaced by the two-digit minute, eg 11.

#endif
*/

DECPHON.C

```

/*-----
MODULE: decphon.c

USE: Decoding phone number from phone.

STORED: Phone number is stored using a stupid formula consisting of
        6 bytes.

Written By: Greg McGregor
GMM 1991

REVISED:          What was Revised?
- GMM 7-30-1991    Nothing
- GMM 8-10-1991    Fixed Wild Pointer , phone number too long gene
rates
                  Jump to address 0x0000 base memory BAD
-----*/

#define TRUE 1
#define FALSE 0

#include <stdio.h>
#include <gkeys.h>
#include <rtbfunc.h>
#include <decphone.h>

#define bit0 0x01
#define bit1 0x02
#define bit2 0x04
#define bit3 0x08
#define bit4 0x10
#define bit5 0x20
#define bit6 0x40
#define bit7 0x80

/*
 *
main () {
char phone [20];
char data [20];

    data [0] = 0x0D;
    data [1] = 0x20;
    data [2] = 0x05;
    data [3] = 0x65;
    data [4] = 0xF1;
    data [5] = 0x50;

    decode_phone (phone,data);
}
*/

```


DECPHON.C

```

/*-----
// Function Name -> decode_phone
// Parameters:
// Function: Decode the phone number of the phone
// Returns:
// Written By : Greg McGregor
//
-----*/

```

```

int decode_phone (char *dest, char *src) {
char area_code [10]; /* 10's to dissallow NMI problems */
char prefix [10];
char digits [10];
unsigned char bytes[10];

```

```

    bytes [0] = src[0];
    bytes [1] = src[1];
    get_area_code (area_code, bytes);

```

```

    bytes [0] = src[2];
    bytes [1] = src[3];
    get_prefix (prefix, bytes);

```

```

    bytes [0] = src[3];
    bytes [1] = src[4];
    bytes [2] = src[5];
    get_digits (digits, bytes);
    sprintf (dest, "%s-%s-%s", area_code, prefix, digits);
    return TRUE;
}

```

```

/*-----
final_decode : every 0 is counted as 10, 100, or 1000 depending
on it's place. after every op you must check next most sig
number for = to 0. if so sub place value. (PAIN IN ASS)
-----*/

```

```

final_decode (int *n) {
int x, sav;
    x = *n;
    sav = x;
    if (x >= 1000) {
        if (x % 10 == 0)
            sav = sav - 10;
        x = sav; /* created a new number, check this for zeros */
        x = x / 10; /* adjust place value */
        if (x % 10 == 0)
            sav = sav - 100;
        x = sav;
        x = x / 100; /* look at hundredths place */
        if (x % 10 == 0)
            sav = sav - 1000;
    } else
        if (x >= 100) {
            if (x % 10 == 0)

```


DECPHON.C

```

        sav = sav - 10;
        x = sav;
        x = x / 10;
        if (x % 10 == 0)
            sav = sav - 100;
    } else
    if (x >= 10) {
        if (x % 10 == 0)
            sav = sav - 10;
    }
    *n = sav;
}

/*-----
get_digits
-----*/
get_digits (char *dest, char *src) {
    unsigned char c, c1;
    unsigned char first_digit;
    unsigned int n;
    unsigned char bytes[10];

    c = src[0];
    c1 = src [1];
    c = c & 0x03; /* nuke top 6 bits */
    c = c << 2; /* shift left 2 */
    c = c & 0x0C; /* nuke bottom 2 bits if needed */
    /* copy next fields upper 2 bits to variable 'c' */
    if (c1 & bit7)
        c = c | bit1;
    if (c1 & bit6)
        c = c | bit0;
    /* 1st digit is now in BCD */
    if (c == 0x0A)
        c = 0; /* a = 0 in BCD */

    first_digit = c + '0';

    /* now get other 3 digits */
    c = src [1]; /* byte 2 */
    c = c & 0x3F; /* nuke top 2 bits of field */
    src [1] = c;
    bytes [0] = src[2];
    bytes [1] = src[1];

    Xcopy (&n, bytes, 2);
    n = n >> 4;
    n = n & 0x0FFF; /* Nuke upper 4 bits */
    n = n + 111; /* adjust for formula */
    final_decode (&n);
    if ( n < 100 ) {
        sprintf ( dest, "%c0%d", first_digit, n);
    } else

```


DECPHON.C

```

    if ( n < 10 ) {
        sprintf ( dest, "%c00%d", first_digit, n);
    } else sprintf (dest, "%c%d", first_digit, n);
    return;
}

/*-----
get_prefix
-----*/
get_prefix (char *dest, char *src) {
    unsigned int n;
    unsigned char bytes [10];

    n = 0;
    bytes [1] = src [0];
    bytes[0] = src[1];
    Xcopy (&n, bytes, 2);
    n = n >> 2;
    n = n & 0x3FFF; /* erase upper 2 bits */
    /* prefix is summed.. ex.. (415) = number 304 in dec */

    n = n + 111; /* part of formula to decode */
    /* now dec = 415 as ex... */
    final_decode (&n);
    sprintf (dest, "%d", n);
    return;
}

/*-----
get_area_code
-----*/
get_area_code (char *dest, char *src) {
    unsigned int n;
    unsigned char bytes [10];

    n = 0;
    bytes [1] = src [0];
    bytes[0] = src[1];
    Xcopy (&n, bytes, 2);
    n = n >> 4;
    n = n & 0x0FFF; /* area code is summed.. ex.. (415) = number 304
in dec */

    n = n + 111; /* part of formula to decode */
    /* now dec = 415 as ex... */
    final_decode (&n);
    sprintf (dest, "%d", n);
    return;
}

Xcopy (char *dest, char *src, int len) {
    int i;
    for (i=0; i<len; i++)

```


DECPHON.C

```
        dest[i] = src [i];  
return;
```

```
}
```


DETAIL.C

```

/*-----
--
MODULE: detail.c

ENTRY POINT: show_detail ()
PURPOSE: Show call record detail with adjustments taxes etc...

Written By : Greg McGregor 1990

REVISED:          What was revised?
- GMM 7-30-1991    Nothing
-----
*/

#include <stdio.h>
#include <gkeys.h>
#include <windows.h>
#include <time.h>
#include <bench.h>
#include <proc.io>
#include <gbase.h>
#include <agrio.h>
#include <agreev3.h>
#include <extnvar.h>
#include <extscrns.h>

/*-----
--
show_detail ()
-----*
/
show_detail ()
{
    detail_wt = windowopen (&detail_win);
    setttitle (detail_wt, "Detailed Billing Screen", CenterUpperTitle);
    dcommands_wt = windowopen (&dcommands_win);
    setttitle (dcommands_wt, "Commands", CenterUpperTitle);
    show_commands ();
    use (detail_wt);
    show_detail_scr ();
    do_detail_entry ();
    windowclose (detail_wt);
    windowclose (dcommands_wt);
}

/*-----
--
show_detail_scr
-----*
/
show_detail_scr ()
{
    use (detail_wt);
    clrscr ();
    gotoxy (2,2);

```


DETAIL.C

```

cprintf ("Rental Date : ");
cprintf ("%s",agreemntrec.rentaldate);
gotoxy (25,2);
cprintf ("Returned Date : ");
cprintf ("%s",agreemntrec.actrtndate);
gotoxy (2,3);
cprintf ("Rental Time : ");
cprintf ("%s",agreemntrec.timeout);
gotoxy (25,3);
cprintf ("Returned Time : ");
cprintf ("%s",agreemntrec.timein);
gotoxy (2,5);
cprintf ("Days Used : ");
cprintf ("%2.0f",agreemntrec.daysused);
gotoxy (15,7);
cprintf ("Days Usage Charge          : ");
cprintf ("%4.2f",agreemntrec.dlyphochg);
gotoxy (15,8);
cprintf ("Phone Usage Charge          : ");
cprintf ("%4.2f",agreemntrec.minphochg);
gotoxy (15,10);
cprintf ("Unreturned Equip. Charge : ");
cprintf ("%4.2f",agreemntrec.equipchg);
gotoxy (15,11);
cprintf ("Adjustments                  :");
cprintf ("<%4.2f>",agreemntrec.adjustment);
gotoxy (15,12);
cprintf ("Discount %                  : ");
cprintf ("%3.0f",agreemntrec.discount);
gotoxy (15,13);
cprintf ("Subtotal                    : ");
cprintf ("%4.2f",agreemntrec.subtotal);
gotoxy (15,15);
cprintf ("Total Tax                   : ");
cprintf ("%3.2f",agreemntrec.total_tax);
gotoxy (15,16);
cprintf ("-----");
gotoxy (15,17);
cprintf ("Net Due                      : ");
cprintf ("%5.2f",agreemntrec.netdue);
}

```

```

/*-----
show_call_listing ()
-----*/
show_call_listing ()
{
int calls,i,x,y;
record_type *a_call;
int min,max,P_continue;
char c,key;
int page,total_pages;

page = 1;

```


DETAIL:C

```
P_continue = TRUE;
x = 1;
y = 3;
calls = call_rec.attached_records;
max = calls;
min = 1;
if (calls > 10) max = 10;
while (P_continue) {
    use (detail_wt);
    clrscr ();
    gotoxy (1,2);
    cprintf ("CALLED          Date      Time      Length Local Long D.
Total");
    x = 1;
    y = 4;
    for (i=min;i<=max;i++ ) {
        a_call = g_get_call (call_rec,i);
        gotoxy (x,y);
        cprintf ("%s",a_call->number);
        gotoxy (14,y);
        cprintf ("%s",a_call->date);
        gotoxy (21,y);
        cprintf ("%s",a_call->start_time);
        gotoxy (33,y);
        cprintf ("%d",(int)a_call->length);
        gotoxy (37,y);
        cprintf ("%4.2f",a_call->base_cost);
        gotoxy (44,y);
        cprintf ("%4.2f",a_call->long_dist_cost);
        gotoxy (51,y);
        cprintf ("%4.2f",a_call->total_cost);
        ++y;
    }
    if (max == calls) {
        gotoxy (51,y);
        cprintf ("=====");
        gotoxy (35,++y);
        cprintf ("TOTAL : ");
        gotoxy (51,y);
        cprintf ("%4.2f",total_rtb_bill);
    } else {
        gotoxy (43,y+1);
        total_pages = calls/10;
        if (calls % 10) ++total_pages;
        cprintf ("Page %d of %d",page,total_pages);
    }
    c = getch();
    if (is_extended_key (c,&key)) {
        if (key == K_F6) return;
        if (key == K_F3) {
            if (min >= 1) {
                if ( (min - 10) >= 1){
                    min -= 10;
                    --page;
                }
            }
        }
    }
}
```


DETAIL.C

```

= calls;

} else {
    min = 1;
    page = 1;
}
max = min + 10;
if ( (max >= calls) ) max

}
}
if (key == K_F4) {
    if (max <= calls) {
        if ( (max + 10) > calls) {
            max = calls;
            page = total_pages;
        } else {
            max += 10;
            ++page;
        }
        if ( (max - 10) <= 1) {
            min = 1;
        } else min = max - 10;
    }
}

}

}

/*-----
show_commands ()
-----*/
show_commands ()
{
    use (dcommands_wt);
    gotoxy (1,3);
    cprintf ("F1 - Call List");
    gotoxy (1,7);
    cprintf ("F3 - Browse Up");
    gotoxy (1,9);
    cprintf ("F4 - Browse Down");
    gotoxy (1,13);
    cprintf ("F6 - Exit");
}

/*-----
do_detail_entry ()
-----*/
do_detail_entry ()
{
    int FIELD,done,key;

```


DETAIL.C

```

    derive_other ();
    show_detail_scr ();
    FIELD = 1;
    done = FALSE;

    while (!done ){
switch (FIELD) {
    case 1: key = get_adjustments_end ();
            break;
    case 2: key = get_discount_detail_end ();
            break;
}
    if (key == K_F1) {
        show_call_listing ();
        use (detail_wt);
        clrscr ();
        show_detail_scr ();
    }
    if ( (key == K_F3) || (key == K_F4) ){
        strcpy (errmsg, "Must Be in Call List
Mode, F1 key");

        errrtn (errmsg);
        use (detail_wt);
        textcolor (White); /* not good, but have
to reset screen */

        textbackground (Black);
        clrscr ();
        show_detail_scr ();
    }
    if (UP_FIELD) {
        if (FIELD == 1) {
            FIELD = 2;
        } else FIELD = 1;
    }
    if (DOWN_FIELD) {
        if (FIELD == 1) {
            FIELD = 2;
        } else FIELD = 1;
    }
    if (key == K_F6) {
        return;
    }

    /* do calc's and etc... */
    derive_other ();
    use (detail_wt);
    show_detail_scr ();

} /* while loop end */
}

```


DISPEND.C

```

/*-----
-
MODULE: dispend.c
PURPOSE: Display MODLUE endagr screens
Written By : Greg McGregor 1990

REVISED:                What was revised?
- GMM 7-30-1991          Nothing
-----*

```

```

#include <stdio.h>
#include <windows.h>
#include <bench.h>
#include <proc.io>
#include <time.h>
#include <agrio.h>
#include <gbase.h>
#include <gkeys.h>
#include <getline.h>
#include <agreev3.h>
#include <extnvar.h>
#include <extscrns.h>
#include <getline.h>

```

```

display_rtb_charges_end ()
{
    use (data_wt_end);
    gotoxy (62,6);
    cprintf ("%4.2f",agreemntrec.minphochg);
}

```

```

display_days_charge_end ()
{
    use (data_wt_end);
    gotoxy (62,5);
    cprintf ("          ");
    gotoxy (62,5);
    cprintf ("%4.2f",agreemntrec.dlyphochg);
}

```

```

display_credit_info ()
{
    capAdjust (agreemntrec.custname,25);
    display_card_name_end ();
    display_card_number_end ();
    display_card_expr_end ();
}

```

```

display_card_name_end ()

```


DISPEND.C

```
{
int i;

    use(data_wt_end);
    gotoxy (20,2);
    textbackground (Black);
    if (strlen(agreementrec.custname) < 24)
        for (i=1;i<=24;i++)
            cprintf (" ");
    gotoxy (20,2);
    cprintfN (agreementrec.custname,24);
}

int get_card_name_end ()
{
    return get_line
(agreementrec.custname,5,2,24,data_wt_end,"Customer Name: ");
}

display_card_number_end ()
{
int i;

    use (data_wt_end);
    gotoxy (20,3);
    textbackground (Black);
    if (strlen(agreementrec.creditno) < 16)
        for (i=1;i<=16;i++)
            cprintf (" ");
    gotoxy (20,3);
    cprintfN (agreementrec.creditno,16);
}

int get_card_number_end ()
{
    return get_line (agreementrec.creditno,5,3,16,data_wt_end,
"Card Number : ");
}

display_card_expr_end ()
{
    use (data_wt_end);
    gotoxy (20,4);
    cprintfN (agreementrec.expiredate,4);
}

int get_card_expr_end ()
{
    return get_line
(agreementrec.expiredate,5,4,4,data_wt_end,"Card Expr. : ");
}

display_phone_number_end ()
{
    use (data_wt_end);
    gotoxy (59,2);
}
```


DISPEND.C

```

        cprintfN (agreemntrec.curphoneno,12);
    }

display_agreement_end()
{
    use (data_wt_end);
    gotoxy (59,3);
    cprintfN (agreemntrec.agreeno,13);
}

display_batteries_end()
{
    int t;

    use (data_wt_end);
    gotoxy (26,9);
    t = agreemntrec.nobatrttn;
    cprintf ("%d",t);
}

display_batteries_rented_end ()
{
    int t;

    use (data_wt_end);
    gotoxy (33,9);
    t = agreemntrec.nobatrent;
    cprintf ("%d",t);
}

int get_batteries_end()
{
    char s[20];
    int stat,ok;
    wintype win;
    ok = FALSE;
    while (!ok) {
        itoa (agreemntrec.nobatrttn,s,10);
        stat = get_line (s,5,9,2,data_wt_end,"No. Extra Batteries
:");
        if (stat == K_CARD_READER) display_credit_info ();
        while (!isdigit (s[0])) {
            if (!isdigit (s[0])) {
                strcpy (errmsg,"Must Be Numeric 0 - 9");
                errrtn(errmessage);
            }
            stat = get_line (s,5,9,2,data_wt_end,"No. Extra
Batteries :");
            if (stat == K_CARD_READER) display_credit_info ();
        }
        agreemntrec.nobatrttn = atof (s);
        if (agreemntrec.nobatrttn > agreemntrec.nobatrent) {
            strcpy (errmsg,"Can't Return More Batteries
Than Rented!");
        }
    }
}

```


DISPEND.C

```

        errrtn (errmsg);
    } else ok = TRUE;
}

return stat;
}

display_chargers_end()
{
    int t;

    use(data_wt_end);
    gotoxy (26,8);
    t = agreemntrec.nochgrtn;
    cprintf ("%d",t);
}

display_chargers_rented_end ()
{
    int t;

    use (data_wt_end);
    gotoxy (33,8);
    t = agreemntrec.nochgrent;
    cprintf ("%d",t);
}

int get_chargers_end()
{
    char s[20];
    int stat,ok;
    wintype win;
    ok = FALSE;
    while (!ok) {
        itoa (agreemntrec.nochgrtn,s,10);
        stat = get_line (s,5,8,2,data_wt_end,"No. Chargers
:");

        if (stat == K_CARD_READER) display_credit_info ();
        while (!isdigit (s[0])) {
            if (!isdigit (s[0])) {
                strcpy (errmsg,"Must Be Numeric 0 -
9");
                errrtn(errmsg);
            }
            stat = get_line (s,5,8,2,data_wt_end,"No. Chargers
:");

            if (stat == K_CARD_READER) display_credit_info ();
        }
        agreemntrec.nochgrtn = atof (s);
        if (agreemntrec.nochgrtn > agreemntrec.nochgrent) {
            strcpy (errmsg,"Can't Return More Chargers
Than Rented!");
            errrtn (errmsg);
        } else ok = TRUE;
    }

    return stat;
}

```


DISPEND.C

```
display_discount_end ()
```

```
{
int t;

        use (data_wt_end);
        gotoxy (18,11);
        t = agreemntrec.discount;
        cprintf ("%d",t);
}
```

```
int get_discount_end()
```

```
{
char s[20];
int stat,in_range;
wintype win;
        in_range = FALSE;
        itoa (agreemntrec.discount,s,10);
        stat = get_line (s,5,11,3,data_wt_end,"Discount % : ");
        if (stat == K_CARD_READER) display_credit_info ();
        while ( (!isdigit (s[0])) ) {
                if (!isdigit (s[0])) {
                        strcpy (errmessage,"Must Be Numeric 0 - 9");
                        errrtn(errmessage);
                }
                stat = get_line (s,5,11,3,data_wt_end,"Discount % : ");
                if (stat == K_CARD_READER) display_credit_info ();
        }
        if ( (atof(s) >=0) && (atof(s) <=100) )
                in_range = TRUE;
        while ( !in_range ) {
                if (!in_range) {
                        strcpy (errmessage,"Must Be A Percent 0 - 100");
                        errrtn(errmessage);
                }
                stat = get_line (s,5,11,3,data_wt_end,"Discount % : ");
                if (stat == K_CARD_READER) display_credit_info ();
                if ( (atof(s) >=0) && (atof(s) <=100) )
                        in_range = TRUE;
        }
        agreemntrec.discount = atof (s);
        return stat;
}
```

```
get_discount_detail_end ()
```

```
{
char s[20];
int stat,in_range;
wintype win;
        in_range = FALSE;
        itoa (agreemntrec.discount,s,10);
        stat = get_line (s,15,12,3,detail_wt,"Discount % : ");
}
```


DISPEND.C

```

        while ( (!isdigit (s[0])) ) {
            if (!isdigit (s[0])) {
                strcpy (errmsgage,"Must Be Numeric 0 - 9");
                errrtn(errmessage);
            }
            stat = get_line (s,15,12,3,detail_wt,"Discount %
: ");
        }
    if ( (atof(s) >=0) && (atof(s) <=100) )
        in_range = TRUE;
    while ( !in_range ) {
        if (!in_range) {
            strcpy (errmsgage,"Must Be A Percent 0 - 100");
            errrtn(errmessage);
        }
        stat = get_line (s,15,12,3,detail_wt,"Discount %
: ");
        if ( (atof(s) >=0) && (atof(s) <=100) )
            in_range = TRUE;
    }
    agreemntrec.discount = atof (s);
    return stat;
}

display_remarks1_end()
{
    gotoxy (5,2);
    cprintfN (agreemntrec.remarks1,34);
}

get_remarks1_end ()
{
    /* null out field if nothing in it */
    if (agreemntrec.remarks1[0] == ' ') agreemntrec.remarks1[0] = '\0';
    return get_line (agreemntrec.remarks1,5,2,34,remarks_wt,"");
}

display_remarks2_end ()
{
    gotoxy (5,3);
    cprintfN (agreemntrec.remarks2,34);
}

get_remarks2_end ()
{
    /* null out field if nothing in it */
    if (agreemntrec.remarks2[0] == ' ') agreemntrec.remarks2[0] = '\0';
    return get_line (agreemntrec.remarks2,5,3,34,remarks_wt,"");
}

display_remarks3_end()
{

```


DISPEND.C

```

    gotoxy (5,4);
    cprintfN (agreemntrec.remarks3,34);
}

get_remarks3_end ()
{
    /* null out field if nothing in it */
    if (agreemntrec.remarks3[0] == ' ') agreemntrec.remarks3[0] = '\0';
    return get_line (agreemntrec.remarks3,5,4,34,remarks_wt,"");
}

display_adjustments_end ()
{
    use (detail_wt);
    gotoxy (42,11);
    cprintf ("<%4.2f",agreemntrec.adjustment);
    gotoxy (49,11);
    cprintf (">");
}

get_adjustments_end ()
{
    char s[20],s1[20];
    int stat,ok;
    wintype win;
    float t,t2;

    null_field (s,20);
    ok = FALSE;
    while (!ok) {
        gcvt (agreemntrec.adjustment,7,s);
        null_field (s1,20); /* make sure field is nulled or - */
        strcpy (s1,s); /* conversion problems occur */
        null_field (s,20);
        strcpy (s,s1);
        gotoxy (49,11);
        cprintf (">");
        stat = get_line (s,15,11,7,detail_wt,"Adjustments
: <");

        while ( (!isdigit (s[0])) && (s[0] != '-') ) {
            if ( (!isdigit (s[0])) && (s[0] != '-') ){
                strcpy (errmsg,"Must Be Numeric 0 -
9");

                errrtn(errmessage);
            }
            gotoxy (49,11);
            cprintf (">");
            stat = get_line (s,15,11,7,detail_wt,"Adjustments
: <");
        }
        t = atof (s);
        t2 = agreemntrec.adjustment; /* previous adjustment */
    }
}

```


DISPEND.C

```
        t2 = ( t * -1.0 ) + ( agreemntrec.subtotal + t2 );
        if ( t2 < 0.0 ) {
            strcpy (errmessage,"Adjustment Too Much!");
            errrtn (errmessage);
        } else ok = TRUE;
    }
    agreemntrec.adjustment = t;
    return ( stat );
}

display_totaltax_end ()
{
    use (data_wt_end);
    gotoxy (62,9);
    cprintf ( "          ");
    gotoxy (62,9);
    cprintf ("%4.2f",agreemntrec.total_tax);
}

/*
// note ldw_charges are added in other for only this display
// they are not added in reality.
*/
display_other_end ()
{
    use (data_wt_end);
    gotoxy (62,7);
    cprintf ( "          ");
    gotoxy (62,7);
    cprintf ("%4.2f", ( other_charges + agreemntrec.ldw_charges ));
}

display_subtotal_end ()
{
    use (data_wt_end);
    gotoxy (62,8);
    cprintf ( "          ");
    gotoxy (62,8);
    cprintf ("%4.2f",agreemntrec.subtotal);
}

display_netdue_end ()
{
    use (data_wt_end);
    gotoxy (62,11);
    cprintf ( "          ");
    gotoxy (62,11);
    cprintf ("%5.2f",agreemntrec.netdue);
}
```


DISPOPEN.C

```

/*-----
-
MODULE: Dispopen.c  initial agreemnt
                Displays various fields on the screen.  Used for editing.
Written By : Greg McGregor

REVISED:          What was revised?
- GMM 7-30-1991    Nothing
GMM 9-5-1991      LDW added
-----
*/

#include <stdio.h>
#include <ctype.h>
#include <gkeys.h>
#include <windows.h>
#include <time.h>
#include <getline.h>
#include <gbase.h>
#include <extnvar.h>
#include <bench.h>
#include <proc.io>
#include <agreev3.h>
#include <agrio.h>
#include <misc.h>
#include <dos.h>

display_card_info_open (wintype a_window)
{
    capAdjust (agreemntrec.custname,25);
    display_card_name_open (a_window);
    display_card_expr_open (a_window);
    display_card_number_open (a_window);
}

display_card_name_open (wintype a_window)
{
    int i;

        use(a_window);
    gotoxy (20,2);
    textbackground (Black);
    if (strlen(agreemntrec.custname) < 24)
        for (i=1;i<=24;i++)
            cprintf (" ");
    gotoxy (20,2);
    cprintf ("%s",agreemntrec.custname);
}

int get_card_name_open (wintype a_window)
{
    int stat;

        stat = get_line

```


DISOPEN.C

```

(agreementrec.custname,5,2,24,a_window,"Customer Name: ");
    if (stat == K_CARD_READER) display_card_info_open
(a_window);
    return stat;
}

display_card_number_open (wintype a_window)
{
    int i;
        use (a_window);
        gotoxy (20,3);
        textbackground (Black);
        if (strlen(agreementrec.creditno) < 16)
            for (i=1;i<=16;i++)
                cprintf (" ");
        gotoxy (20,3);
        cprintf ("%s",agreementrec.creditno);
}

int get_card_number_open (wintype a_window)
{
    int key;

        key = get_line (agreementrec.creditno,5,3,16,a_window,"Card
Number : ");
        switch (agreementrec.creditno[0]){
            case '3' : if (agreementrec.creditno[1] == '7')
                        strcpy
(agreementrec.creditttype,"AE");
                        if
(agreementrec.creditttype[1] == '8')
                        strcpy
(agreementrec.creditttype,"DC");
                        break;
            case '4' : strcpy (agreementrec.creditttype,"VI");
                        break;
            case '5' : strcpy (agreementrec.creditttype,"MC");
                        break;
            case '6' : strcpy (agreementrec.creditttype,"DI");
                        break;
            case '9' : strcpy (agreementrec.creditttype,"CB");
                        break;
        }
        if (key == K_CARD_READER) display_card_info_open (a_window)
        return key;
}

display_card_expr_open (wintype a_window)
{
        use (a_window);
        gotoxy (20,4);
        cprintf ("%s",agreementrec.expiredate);
}

```


DISOPEN.C

```
int get_card_expr_open (wintype a_window)
{
    int stat;
        stat = get_line
(agreemntrec.expiredate,5,4,4,a_window,"Expires      : ");
        if (stat == K_CARD_READER) display_card_info_open
(a_window);
        return stat;
}

display_phone_number_open (wintype a_window)
{
        use (a_window);
    gotoxy (57,3);
    cprintf ("%s",agreemntrec.curphoneno);
}

display_agreement_open (wintype a_window)
{
        use (a_window);
    gotoxy (57,4);
    cprintf ("%s",agreemntrec.agreeno);
}

display_meter_hours_open (wintype a_window)
{
    int t;
        use (a_window);
    gotoxy (62,3);
    t = agreemntrec.hoursout;
    cprintf ("%d",t);
}

display_meter_mins_open (wintype a_window)
{
    int t;
        use (a_window);
    gotoxy (62,4);
    t = agreemntrec.minutesout;
    cprintf ("%d",t);
}

display_drivers_open (wintype a_window)
{
        use (a_window);
    gotoxy (22,6);
    cprintf ("%s",agreemntrec.licenseno);
}

int get_drivers_open (wintype a_window)
{
    int stat;
        stat = get_line
```


DISPOPEN.C

```
        if (stat == K_CARD_READER) display_card_info_open
(a_window);
        return stat;
}

display_address_open (wintype a_window)
{
        use (a_window);
        gotoxy (15,7);
        cprintf ("%s",agreemntrec.custaddr1);
}

int get_address_open (wintype a_window)
{
int stat;

        stat = get_line
(agreemntrec.custaddr1,5,7,23,a_window,"Address : ");
        if (stat == K_CARD_READER) display_card_info_open
(a_window);
        return stat;
}

display_city_open (wintype a_window)
{
        use (a_window);
        gotoxy (15,8);
        cprintf ("%s",agreemntrec.custcity);
}

int get_city_open (wintype a_window)
{
int stat;

        stat = get_line (agreemntrec.custcity,5,8,23,a_window,"City
: ");
        if (stat == K_CARD_READER) display_card_info_open
(a_window);
        return stat;
}

display_state_open (wintype a_window)
{
        use (a_window);
        gotoxy (14,9);
        cprintfN (agreemntrec.custstate,2);
}

display_zip_open (wintype a_window)
{
        use(a_window);
        gotoxy (17,9);
        cprintf ("%s",agreemntrec.custzipcd);
}

int get_state_open (wintype a_window)
```


DISOPEN.C

```

{
char temp[10];
int stat;
    moveX (temp,agreemntrec.custstate,2);
    temp[2] = '\0';
    stat = get_line (temp,5,9,2,a_window,"St/Zip  :");
    moveX (agreemntrec.custstate,temp,2);
    if (stat == K_CARD_READER) display_card_info_open (a_window);
    return stat;
}

int get_zip_open (wintype a_window)
{
char temp[20];
int stat;
    moveX (temp,agreemntrec.custzipcd,9);
    temp [9] = '\0';
    stat = get_line (temp,16,9,9,a_window,"  ");
    moveX (agreemntrec.custzipcd,temp,9);
    if (stat == K_CARD_READER) display_card_info_open (a_window);
    return stat;
}

display_home_phone_open (wintype a_window)
{
    use(a_window);
    gotoxy (17,10);
    cprintfN (agreemntrec.homephone,12);
}

int get_home_phone_open (wintype a_window)
{
char temp[20];
int stat;
    null_field (temp,20);
    if ( (agreemntrec.homephone[0] == ' ') ||
(agreemntrec.homephone[0] == '\0') ){
        moveX (temp," - - -",12);
    }else moveX (temp,agreemntrec.homephone,12);
    temp[12] = '\0';
    stat = get_line_mask (temp,5,10,12,a_window,"Home Phone : ",
- - -");
    moveX (agreemntrec.homephone,temp,12);
    if (stat == K_CARD_READER) display_card_info_open (a_window);
    return stat;
}

display_local_phone_open (wintype a_window)
{
    use(a_window);
    gotoxy (17,11);
    cprintfN (agreemntrec.local_phone_number,12);
}

```


DISPOPEN.C

```

int get_local_phone_open (wintype a_window)
{
char temp[20];
int stat;
    null_field (temp,20);
    if ( (agreemntrec.local_phone_number[0] == ' ') ||
(agreemntrec.local_phone_number[0] == '\0') ){
        moveX (temp,"      ",12);
    } else moveX (temp,agreemntrec.local_phone_number,12);
    temp[12] = '\0';
    stat = get_line_mask (temp,5,11,12,a_window,"Local Phone: ",
- - - - -
    );
    moveX (agreemntrec.local_phone_number,temp,12);
    if (stat == K_CARD_READER) display_card_info_open (a_window);
    return stat;
}

display_estimated_return_date (wintype a_window)
{
char temp[20],t[20];
    null_field (temp,20);
    null_field (t,20);
    if (agreemntrec.estimated_return_date == ' ') {
        moveX (t,"      ",6);
    } else moveX (t,agreemntrec.estimated_return_date,6);
    if (agreemntrec.estimated_return_date[0] == '\0')
        moveX (t,"      ",6);
    temp[6] = t[0]; /* put mask and convert from YYMMDD to MM/DD/YY */
    temp[7] = t[1];
    temp[5] = '/';
    temp[0] = t[2];
    temp[1] = t[3];
    temp[2] = '/';
    temp[3] = t[4];
    temp[4] = t[5];
    temp[8] = '\0';
    use(a_window);
    gotoxy (17,12);
    cprintfN (temp,8);
}

/*-----
// Function Name -> check_return_date
// Parameters:
// Function: TRUE/ FALSE
// Returns:
// Written By : Greg McGregor
//
-----*/

int check_return_date (char *dt) {
char t[10];

```


DISPOPEN.C

```

int i;
time_t computers_time;
struct tm *computers_date;
char str[80];

computers_time = time ( NULL );
computers_date = localtime ( &computers_time );

null_field (t,10);
if ( (!isdigit (dt[6])) || ( !isdigit (dt[7])) ){
    errrtn ( "You must enter digits between 0 - 9 " );
    return ( FALSE );
}
t[0] = dt[6];
t[1] = dt[7];
i = atoi ( t );
if ( computers_date->tm_year > i ) {
    sprintf (str,"The year must be equal or greater than
%d",computers_date->tm_year );
    errrtn ( str );
    return ( FALSE );
}
t[0] = dt[0];
t[1] = dt[1];
i = atoi (t);
if ( (i < 1) || (i > 12) ) {
    errrtn ( "The month must be between 1 and 12");
    return ( FALSE );
}
t[0] = dt[3];
t[1] = dt[4];
i = atoi (t);
if ( (i < 1) || (i > 31) ) {
    errrtn ( "The day must be between 1 and 31");
    return ( FALSE );
}
return TRUE;
}

int get_estimated_return_date (wintype a_window)
{
char temp[20],t[20];
int stat,result;
moveX (t,agreemntrec.estimated_return_date,6);
if (agreemntrec.estimated_return_date[0] == '\0') {
    get_curdate ( &t ); /* plug in current date */
}
temp[6] = t[0]; /* put mask and convert from YYMMDD to MM/DD/YY */
temp[7] = t[1];
temp[5] = '/';
temp[0] = t[2];
temp[1] = t[3];

```


DISPOPEN.C

```

temp[2] = '/';
temp[3] = t[4];
temp[4] = t[5];
temp[8] = '\0';

do {
    stat = get_line_mask (temp,5,12,8,a_window,"Return Date:
", " / / ");
    result = check_return_date (temp);
} while (!result);

null_field (t,8);
t[0] = temp[6]; /* convert from MM/DD/YY to YYMMDD */
t[1] = temp[7];
t[2] = temp[0];
t[3] = temp[1];
t[4] = temp[3];
t[5] = temp[4];
t[6] = '\0';
moveX (agreemntrec.estimated_return_date,t,6);
if (stat == K_CARD_READER) display_card_info_open (a_window);

return ( stat );
}

/*
display_company_open (wintype a_window)
{
    use(a_window);
    gotoxy (17,11);
    cprintf ("%s",agreemntrec.company);
}

int get_company_open (wintype a_window)
{
    int stat;
    stat = get_line (agreemntrec.company,5,11,24,a_window,"Company
");
    if (stat == K_CARD_READER) display_card_info_open (a_window);
    return stat;
}
*/

display_batteries_open (wintype a_window)
{
    int t;
    use (a_window);
    gotoxy (65,8);
    t = agreemntrec.nobatrent;
    cprintf ("%d",t);
}

int get_batteries_open (wintype a_window)

```


DISPOPEN.C

```

{
char s[20];
int stat;
wintype win;

    null_field (s,20);
    itoa (agreemntrec.nobatrent,s,10);
    stat = get_line (s,44,8,2,a_window,"No. Extra Batteries: ");
    if (stat == K_CARD_READER) display_card_info_open
(a_window);
        while (!isdigit (s[0])) {
            if (!isdigit (s[0])) {
                errrtn("Must Be Numeric 0 - 9");
            }
            stat = get_line (s,44,8,2,a_window,"No. Extra
Batteries: ");
            if (stat == K_CARD_READER) display_card_info_open
(a_window);
        }
        agreemntrec.nobatrent = atof (s);
        return stat;
}

display_chargers_open (wintype a_window)
{
int t;

    use(a_window);
    gotoxy (65,9);
    t = agreemntrec.nochgrent;
    cprintf ("%d",t);
}

int get_chargers_open (wintype a_window)
{
char s[20];
int stat;
wintype win;

    null_field (s,20);
    itoa (agreemntrec.nochgrent,s,10);
    stat = get_line (s,44,9,2,a_window,"No. Chargers : ");
    if (stat == K_CARD_READER) display_card_info_open
(a_window);
        while (!isdigit (s[0])) {
            if (!isdigit (s[0])) {
                errrtn("Must Be Numeric 0 - 9");
            }
            stat = get_line (s,44,9,2,a_window,"No. Chargers
: ");
            if (stat == K_CARD_READER) display_card_info_open
(a_window);
        }
        agreemntrec.nochgrent = atof (s);
        return stat;
}

```


DISPOPEN.C

```

/*
display_cases_open (wintype a_window)
{
    int t;

        use (a_window);
        gotoxy (65,10);
        t = agreemntrec.nocasrent;
        cprintf ("%d",t);
}

int get_cases_open (wintype a_window)
{
    char s[20];
    int stat;
    wintype win;

        null_field (s,20);
        itoa (agreemntrec.nocasrent,s,10);
        stat = get_line (s,44,10,2,a_window,"No. Cases
");
        if (stat == K_CARD_READER) display_card_info_open
(a_window);
        while (!isdigit (s[0])) {
            if (!isdigit (s[0])) {
                errrtn ("Must Be Numeric 0 - 9 ");
            }
            stat = get_line (s,44,10,2,a_window,"No. Cases
: ");
            if (stat == K_CARD_READER) display_card_info_open
(a_window);
        }
        agreemntrec.nocasrent = atof (s);
        return stat;
}
*/

display_ldw_open (wintype a_window)
{
    char t;

        use (a_window);
        gotoxy (65,10);
        t = agreemntrec.remarks5[0];
        cprintf ("%c",t);
}

int get_ldw_open (wintype a_window)
{
    char s[20];
    int stat;
    wintype win;
    s[0] = agreemntrec.remarks5[0];
    stat = get_line (s,44,10,1,a_window,"LDW [Y/N] : ");
    if (stat == K_CARD_READER) display_card_info_open (a_window);
    while ( (s[0] != 'Y') && (s[0] != 'N') ) {

```


DISPOPEN.C

```

        if ( (s[0] != 'Y') && (s[0] != 'N') )
            errrtn ("You must enter a Y to accept or N to
decline.");
        stat = get_line (s,44,10,1,a_window,"LDW  [Y/N]      : "
);
        if (stat == K_CARD_READER) display_card_info_open
(a_window);
    }
    agreemntrec.remarks5[0] = s[0];
    return stat;
}

```

```

display_discount_open (wintype a_window)
{

```

```

    int t;
        use (a_window);
        gotoxy (65,11);
        t = agreemntrec.discount;
        cprintf ("%d",t);
}

```

```

int get_discount_open (wintype a_window)
{

```

```

    char s[20];
    int stat,in_range;
    wintype win;
        in_range = FALSE;
        null_field (s,20);
        itoa (agreemntrec.discount,s,10);
        stat = get_line (s,44,11,3,a_window,"Discount %
");
        if (stat == K_CARD_READER) display_card_info_open
(a_window);
        while ( (!isdigit (s[0])) ) {
            if (!isdigit (s[0])) {
                errrtn ("Must Be Numeric 0 - 9");
            }
            stat = get_line (s,44,11,3,a_window,"Discount %
: ");
            if (stat == K_CARD_READER) display_card_info_open
(a_window);
        }
        if ( (atof(s) >=0) && (atof(s) <=100) )
            in_range = TRUE;
        while ( !in_range ) {
            if (!in_range) {
                errrtn("Must Be A Percent 0 - 100");
            }
            stat = get_line (s,44,11,3,a_window,"Discount %
: ");
            if (stat == K_CARD_READER) display_card_info_open

```


DISPOPEN.C

```
(a_window);  
        if ( (atof(s) >=0) && (atof(s) <=100) )  
            in_range = TRUE;  
    }  
    agreemntrec.discount = atof (s);  
    return stat;  
}
```


ENDAGR.C

```

/*-----
MODULE: endagr.c RTB

PURPOSE: Allows user to close an agreement.

Written By: Greg McGregor
GMM 1990

REVISED:          What was revised?

GMM 7-30-1991      Added federal, state air, state rent, county and
                    city taxes.
-----
--*/

#include <process.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <window.h>
#include <dos.h>
#include <bios.h>
#include <ctype.h>
#include <bench.h>
#include <proc.io>
#include <\sys\stat.h>
#include <math.h>

#include <agrio.h>
#include <agreev3.h> /* all types, making them externs */
#include <control.h>
#include <phone.h>
#include <raperson.h>
#include <gbase.h>
#include <extnvar.h> /* patches global variables as externs */
#include <windows.h>
#include <gkeys.h>
#include <extscrns.h>
#include <whatend.h>
#include <misc.h>
#include <getline.h>
#include <cardrdr.h>
#include <credit.h>
#include <dispopen.h>
#include <printer.h>
#include <detail.h>
#include <realtime.h>
#include <taustat.h>
#include <lostdam.h>

```


ENDAGR.C

```

/*
 * Globals
 */

int IS_PHONE_LOST = FALSE;
float global_amount_paid = 0.0;

/*-----
endagr ()      :      ENTRY POINT  IN/OUT OF MODULE
-----*/
endagr ()
{
    int ok;

        ok = TRUE;
        window (1,1,80,25);
        textbackground (Black);
        clrscr ();
        main_window_end ();
        w_init_end ();      /* init what next */
        init_fields_end ();
/*      init_keys (); */    /* done in mainmneu,init keys for endagr module */
/*      rt_init_databases (); */    /* done in mainmenu,init realtime
billing data bases */

        open_files();
        if (!open_rt_files ()) { /* realtime billing files */
            rtb_error (-1);
            ok = FALSE;
        }

        if (!entry_level_end ()) {
            strcpy (errmsg,"Please enter your ID code correctly
next time!");
            errrtn(errmessage);
            ok = FALSE;
        }

        end_agr_func_options_window ();

        if (ok)
            process_all_end();

        close_all_windows ();
        close_files ();

        close_rt_files ();
        error_wt = windowopen (&error_win);
        setttitle (error_wt,"Garbage Collector",CenterUpperTitle);
        gotoxy (1,2);
        centerPrint (60,"F r e e i n g   M e m o r y");

```


ENDAGR.C

```

garbage_collect (&call_rec); /* garbage collect call_rec */
windowclose (error_wt);
PRINTED_CONTRACT = FALSE;
return;
}

/*-----
process_all
-----*/
process_all_end ()
{
int stat;
    stat = do_cti_end ();
    if (stat == -25) return;
    if (stat){
        stat = load_agreemnt (3); /* load by phone number if cti
worked */
    } else stat = load_agreemnt (1); /* by agreemnt number */
    if (!stat) return; /* couldn't find agreemnt */
    if (!derive_fields_end ()) return;
    do_data_end ();
}

/*-----
*
* Procedure Name: end_agr_func_options_window
* Parameters:
* Function:
* Returns:
*
* Written By: Greg McGregor
-----*/
end_agr_func_options_window ()
{
    funckeys_wt = windowopen (&funckeys_win);
    setttitle (funckeys_wt, "Commands", CenterUpperTitle);
    gotoxy (3,1);
    cprintf ("F2  - Cancel");
    gotoxy (3,2);
    cprintf ("F3  - Finish");
    gotoxy (3,3);
    cprintf ("F7  - View Bill");
    gotoxy (3,4);
    cprintf ("F9  - What Next?");
    gotoxy (3,5);
    cprintf ("F10 - More Options");
    use (main_wt);
}

```


ENDAGR.C

```

/*-----
main_window_end:
-----*/
main_window_end()
{
    main_wt = windowopen (&main_win);
    setttitle (main_wt,"* Returning a Phone *",CenterUpperTitle);
    cursoroff ();
    use (main_wt);
}

/*-----
init_fields_end:
-----*/
init_fields_end()
{
    moveX (agreemntrec.curphoneno,"111-111-1111",12);
    PRINTED_CONTRACT = FALSE;
    CARD_APPROVED = FALSE;
}

/*-----
derive_fields_end
-----*/
int derive_fields_end ()
{
    float t1,t2;
    float t3,t4,t5;
    char temp[10];
    int stat;

    get_curdate (agreemntrec.actrtndate);    /* put rental
date in field */
    get_time (agreemntrec.timein);
    moveX (agreemntrec.origagency,controlrec.tau_id,4);
    agreemntrec.phochgday = controlrec.phone_daily_chg;
    agreemntrec.phochgmin = controlrec.charge_per_minute;
    agreemntrec.minphochg = total_rtb_bill;

    t5 = 0;
    if (strncmp
(agreemntrec.rentaldate,agreemntrec.actrtndate,2) !=0)
        t5 = 365;    /* years are different add twelve to
months place */

    t1 = (float) day_in_year (agreemntrec.rentaldate);
    t2 = (float) day_in_year (agreemntrec.actrtndate);

    t2 = t2 + t5;

    t1 = t2 - t1;    /* days used Calendar */

```


ENDAGR:C

```

    strcpy (temp,agreemntrec.timeout);
    add_seconds (temp);
    t3 = time_to_seconds (temp);
    strcpy (temp,agreemntrec.timein);
    add_seconds (temp);
    t4 = time_to_seconds (temp);

    if (t4 > t3) ++t1; /* if time is over 24 hrs add another
day */

    agreemntrec.daysused = t1;
    stat = reset_file9 (fd_control,&controlrec);
    if (stat < 0) {
        strcpy (errmsg,"Can't Find CONTROL RECORDS");
        errrtn(errmessage);
        return FALSE;
    }

    agreemntrec.dlyphochg = controlrec.phone_daily_chg *
agreemntrec.daysused;
    agreemntrec.adjustment = 0;
    moveX (phonerec.curphoneno,agreemntrec.curphoneno,12);
    stat = exactkey9 (fd_phone,&phonerec);
    if (stat < 0) {
        sprintf (errmsg,"derive_fields: ERROR in phone
file! %d",stat);
        errrtn (errmsg);
        return FALSE;
    }
    return TRUE;
}

/*-----
-
derive_other : derive misc charges, damage , taxes etc...
-----
*/
derive_other ()
{
    agreemntrec.equipchg = 0;

    if (agreemntrec.nobatrent > agreemntrec.nobatrttn) {
        agreemntrec.equipchg += (agreemntrec.nobatrent -
agreemntrec.nobatrttn)
*
controlrec.lost_battery_chg;
    }

    if (agreemntrec.nochgrent > agreemntrec.nochgrtn) {
        agreemntrec.equipchg += (agreemntrec.nochgrent -

```


ENDAGR:C

```
agreemntrec.nochgrtn)
controlrec.lost_charger_chg;
}

    if (agreemntrec.remarks5[0] == 'Y') {
        agreemntrec.ldw_charges = controlrec.ldw_daily_chg *
agreemntrec.daysused;
    } else agreemntrec.ldw_charges = 0.0;

    if ( (IS_PHONE_LOST) && (agreemntrec.remarks5[0] != 'Y') ) /* rem
5 is ldw*/
        agreemntrec.equipchg += controlrec.lost_phone_chg;

/*    round_f (&agreemntrec.equipchg); */

    agreemntrec.dlyphochg = controlrec.phone_daily_chg *
agreemntrec.daysused;
/*    round_f (&agreemntrec.dlyphochg); */

    if (agreemntrec.discount != 0) {
        agreemntrec.dlyphochg = agreemntrec.dlyphochg -
                                ( agreemntrec.discount/100 *
                                agreemntrec.dlyphochg);

/*    round_f (&agreemntrec.dlyphochg); */

    }

    agreemntrec.subtotal = agreemntrec.dlyphochg +
agreemntrec.minphochg +
                                agreemntrec.equipchg + (
agreemntrec.adjustment * -1.0 )+ agreemntrec.ldw_charges;

/*    round_f (&agreemntrec.subtotal); */

/* calculate tax rates and total taxes */

    agreemntrec.federal_tax_air = agreemntrec.minphochg *
controlrec.federal_tax_air;
    agreemntrec.federal_tax_rent = agreemntrec.dlyphochg *
controlrec.federal_tax_rent +
                                agreemntrec.ldw_charges *
controlrec.federal_tax_rent;
    agreemntrec.federal_tax_lost = agreemntrec.equipchg *
controlrec.federal_tax_lost;
    agreemntrec.state_tax_air = agreemntrec.minphochg *
controlrec.state_tax_air;
    agreemntrec.state_tax_rent = agreemntrec.dlyphochg *
controlrec.state_tax_rent +
                                agreemntrec.ldw_charges *
controlrec.state_tax_rent;
    agreemntrec.state_tax_lost = agreemntrec.equipchg *
```


ENDAGR.C

```

/*      round_f (&agreemntrec.total_tax); */

agreemntrec.netdue = agreemntrec.subtotal + agreemntrec.total_tax;

/*      round_f (&agreemntrec.netdue); */

agreemntrec.amtpaid = global_amount_paid;
agreemntrec.amtowed = agreemntrec.netdue - global_amount_paid;

if (agreemntrec.amtowed != 0.0)
    update_tau_status (4, '8');

other_charges = ( agreemntrec.adjustment * -1.0 )+
agreemntrec.equipchg;

strcpy (agreemntrec.preparedby, returned_to);

agreemntrec.phochgday = controlrec.phone_daily_chg;

agreemntrec.calls_made = (float) number_of_calls;
agreemntrec.base_cost = base_cost;
agreemntrec.long_dist = long_dist;

if ( (agreemntrec.nobatrent > agreemntrec.nobatrttn) ||
      (agreemntrec.nochgrent > agreemntrec.nochgrtn) ){
    update_tau_status (1, '4');
} else update_tau_status (1, ' ');
}

/*-----
int day_in_year : Takes YYMMDD
-----*/
int day_in_year (char *d)
{
    int t1,t2;
    int days;

    char temp[5];
    temp[2] = '\0';
    temp[0] = d[2];
    temp[1] = d[3];
    t1 = atoi (temp);
    temp[0] = d[4];
    temp[1] = d[5];
    t2 = atoi (temp);
    days = 0;
    days = add_days (--t1); /* days in months to previous month */
    days = days + t2; /* add in day of up to now */
    return days;
}

```


ENDAGR.C

```

/*-----
add_seconds  add seconds to a time in format HH:MM(A/P) to HH:MM:SS(A/P
)
-----*/
add_seconds (char *s)
{
    s[9] = '\0';
    s[8] = s[5];
    s[5] = ':';
    s[6] = '0';
    s[7] = '0';
}

/*-----
add_days : recursive function to add up days
-----*
/
int add_days (int month)
{
    if (month == 0) {
        return 0;
    } else {
        switch (month) {
            case 1 : return ( 31 + add_days (--month));
                     break;
            case 2 : return ( 28 + add_days (--month));
                     break;
            case 3 : return ( 31 + add_days (--month));
                     break;
            case 4 : return ( 30 + add_days (--month));
                     break;
            case 5 : return ( 31 + add_days (--month));
                     break;
            case 6 : return ( 30 + add_days (--month));
                     break;
            case 7 : return ( 31 + add_days (--month));
                     break;
            case 8 : return ( 31 + add_days (--month));
                     break;
            case 9 : return ( 30 + add_days (--month));
                     break;
            case 10: return ( 31 + add_days (--month));
                     break;
            case 11: return ( 30 + add_days (--month));
                     break;
            case 12: return ( 31 + add_days (--month));
                     break;
        }
    }
}

/*-----
entry_level_end : legitimate employee ?
-----*/

```


ENDAGR.C

```

int entry_level_end ()
{
wintype win,win2;
int key, iostat;
char code[4];
    strcpy (code," ");
    win = windowopen (&entry_win);
    setttitle (win,"Entry Level",CenterUpperTitle);
    cursoron ();
    key = get_line (code,20,1,3,win,"Enter Your ID Code --> ");
    if (key == K_F2) return FALSE;
    fcopy (rapersonrec.rapid, code, 3);
    iostat = exactkey9 (fd_raperson, &rapersonrec);
    windowclose (win);
    if (iostat < 0)
        return FALSE;
    strcpy (returned_to,code);
    return TRUE;
}

/*-----
load_agreemnt - load up an agreemnt
-----*/
int load_agreemnt (int key) /* key = 1 agreeno key = 3 phoneno */
{
int iostat,found;
wintype win2;
char agreeno_save[20];
struct agreemnt_def temp_agreemnt;

    found = FALSE;
    iostat = reset_file9 (fd_agreemnt,&temp_agreemnt);
    if (key == 3) {
        selectinx9 (fd_agreemnt,3);
        iostat = 0;
        iostat = exactkey9(fd_agreemnt, &agreemntrec);
        if (iostat < 0) {
            win2 = note ("Can't Find
Agreement!");
            gotoxy (10,3);
            gotoxy (20,3);
            cprintf ("Press ESC to continue");
            gotoxy (1,4);
            cprintf ("%d",iostat);
            getch();
            windowclose (win2);
        } else found = TRUE;
        do{
            moveX(agreeno_save,agreemntrec.agreeno,13);
            iostat = nextkey9(fd_agreemnt,
            &agreemntrec);
            if (iostat == 0){

```


ENDAGR.C

```

moveX(agreeno_save,agreemntrec.agreeno,13);
    }
    while (iostat == 0);
    selectinx9(fd_agreemnt, 1);    /* read
using agreement number */
    moveX(agreemntrec.agreeno,agreeno_save,13);
    iostat = exactkey9(fd_agreemnt,
&agreemntrec);
    }
    if (found) {
        w_log_end (AGREEMENT_STEP);
        return TRUE;
    }
    return FALSE;
}

/*-----
do_cti_end
-----*/

/
int do_cti_end ()
{
char s[80],s1[80];
gbaserec r; /* a call listing data base structure, gbase.c */
char ch,key;
int done = FALSE;

    CTI_wt = windowopen (&CTI_win);
    settitle (CTI_wt,"Telephone Return",CenterUpperTitle);
    centerPrint (50,"STEP 1 -> Place Phone in CTI");
    note_wt = windowopen (&note_win);
    settitle (note_wt,"CTI Process",CenterUpperTitle);
    gotoxy (1,2);
    centerPrint (60,"Do NOT Remove Phone From CTI!");
    gotoxy (1,3);
    centerPrint (60,"Wait One Moment!");

    end_rtb (); /* do phone check in */

    windowclose (note_wt);

    /* call_rec is global and errors are returned in
attached_records */
    if ( (call_rec.attached_records == -25) ) {
        lost_phone_message ();
        derive_fields_end ();
        derive_other ();
        add_upd_agreemnt (4); /* phone broken, can't communicate
*/

        do_credit_end ();
        if (!CARD_APPROVED) update_tau_status (4,'8');
        if (print_contract (2,FALSE) != 0) {
            errrtn ("Could Not Print Receipt!");

```


ENDAGR.C

```

        update_tau_status (2,'5');
    }
    add_upd_agreemnt (4);
    system ("ccopyit agreemnt");
    system ("ccopyit phone");
    system ("ccopyit callrec.dat");
    return FALSE;
}
if ( (call_rec.attached_records == -29) ){
    lost_phone_message ();
    IS_PHONE_LOST = TRUE;
    derive_fields_end (); /* fill in fields */
    derive_other ();
    do_credit_end ();
    add_upd_agreemnt (3); /* phone lost = 3 */
    if (!CARD_APPROVED) update_tau_status (4,'8');
    if (print_contract (2,TRUE) != 0) {
        errrtn ("Could Not Print Receipt!");
        update_tau_status (2,'5');
    }
    add_upd_agreemnt (3);
    system ("ccopyit agreemnt");
    system ("ccopyit phone");
    system ("ccopyit callrec.dat");
    IS_PHONE_LOST = FALSE; /* reset flag */
    return FALSE;
} else
    if (call_rec.attached_records < 0) {
        sprintf (s,"Sorry - you must start the rental over. (error
%d)",call_rec.attached_records * -1);
        errrtn (s);
        return FALSE;
    }
    w_log_end (CTI_STEP); /* log step done */
    return TRUE;
}

/*-----
do_credit_end:  do credit authorization
-----*/
do_credit_end ()
{
    char s[80],temp[20],ch,response[80];
    char temp_authnumber[80];
    int done,stat,RETRY_CREDIT = FALSE;
    int yesno = FALSE;
    float f;
    wintype win1;

    retry:
        yesno = FALSE;
        agreemntrec.credit_attempted[0] = 'Y';
        if (agreemntrec.efundtrans[0] == 'Y') {
            strcpy (errmsg,"Credit Auth. Already Approved For This

```


263

Page 12

ENDAGR.C

```

Contract!");
    errrtn (errmsg);
} else
if (!w_is_logged_end (CREDIT_STEP)) {
    gotoxy (5,2);
    strcpy (s,"Collecting $");
    sprintf (temp,"%-4.2f",agreemntrec.netdue);
    strcat (s,temp);
    strcat (s,"...Continue ? (Y/N)?");
    error_wt = windowopen (&error_win);
    settitle (error_wt,"Note",CenterUpperTitle);
    beep ();
    gotoxy (5,2);
    if (yes_no (s,TRUE) ) {
        yesno = TRUE;
        windowclose (error_wt);
        note_wt = note ("Wait While Credit
Authorization Is Processed!");
        credit_wt = windowopen (&credit_win);
        settitle (credit_wt,"Credit Card
Authorization",CenterUpperTitle);
        CREDIT_WIN_OPEN = TRUE;
        win1 = windowopen (&card_win);
        settitle (win1,"Authorizing
Card",CenterUpperTitle);

        gotoxy (1,3);
        strcpy (s," Card No: ");
        strcat (s,agreemntrec.creditno);
        cprintf ("%s",s);
        gotoxy (1,2);
        strcpy (s," Name: ");
        strcat (s,agreemntrec.custname);
        cprintf ("%s",s);
        gotoxy (1,4);
        strcpy (s," Expr: ");
        strcat (s,agreemntrec.expiredate);
        cprintf ("%s",s);

        strcpy
(agreemntrec.approved,agreemntrec.preapproved);
        CARD_APPROVED = get_credit

```


ENDAGR.C

response

agreemntrec.approved,

1); /*

0=com1 1=com2 */

```
use (win1);
windowclose (win1);
if (CARD_APPROVED) {
    w_log_end (CREDIT_STEP);
    agreemntrec.efundtrans[0] = 'Y';
    global_amount_paid =
```

agreemntrec.netdue;

```
    }
    use (note_wt);
    windowclose (note_wt);
} else windowclose (error_wt);
} else {
    strcpy (errmsg, "Credit Authorization Already Done!");
    errrtm (errmsg);
}
done = FALSE;
if ( (!CARD_APPROVED) && (yesno) ){
    note_wt = windowopen (&note_win);
    setttitle (note_wt, " Credit Card Message ", CenterUpperTitle);
    gotoxy (1,1);
    centerPrint (60, response);
    gotoxy (1,3);
    centerPrint (60, "Press ESC to Exit or Swipe a Card!");
    while (!done) {
        ch = getch();
        if (ch == '%') {
            ungetch(ch);    /* put back the
```

% */

read_in_card(agreemntrec.creditno,

agreemntrec.custname,

agreemntrec.expireddate,

agreemntrec.creditttype);

capAdjust

(agreemntrec.custname, 24);

shorten_blanks

(agreemntrec.custname);

textbackground (Black);

done = TRUE;

RETRY_CREDIT = TRUE;

```
    }
    if (ch == K_ESC) {
        done = TRUE;
```

```
    }
```

```
}
```


ENDAGR.C

```
        windowclose (note_wt);
        use (credit_wt);
    }
    if (RETRY_CREDIT) goto retry;
}

/*-----
do_data_end
-----*/

/
do_data_end ()
{
    data_wt_end = windowopen (&data_win_end);
    setttitle (data_wt_end, " Data Entry Screen ", CenterUpperTitle);
    data_end ();
}

/*-----
data_end
-----*/

data_end ()
{
    char s[80];
    int FIELD = 1;
    int done;

    cursoron ();
    done = FALSE;
    display_scr1_end ();
    display_values_scr1_end();
    get_data_end ();
}

/*-----
---
display_scr1_end()
-----
-*/
display_scr1_end()
{
    use (data_wt_end);
    gotoxy (5,2);
    cprintf ("Customer Name: ");
    gotoxy (5,3);
    cprintf ("Card Number : ");
    gotoxy (5,4);
```


ENDAGR.C

```

cprintf ("Rented");
gotoxy (30,7);
cprintf ("-----");
gotoxy (5,8);
cprintf ("No. Chargers      :");
gotoxy (5,9);
cprintf ("No. Extra Batteries :");
gotoxy (5,11);
cprintf ("Discount % :");
gotoxy (45,2);
cprintf ("Phone #      :");
gotoxy (45,3);
cprintf ("Agreement # :");
gotoxy (43,4);
cprintf ("-----");
gotoxy (45,5);
cprintf ("Days Charge      :");
gotoxy (45,6);
cprintf ("Phone Usage Chg:");
gotoxy (45,7);
cprintf ("Other            :");
gotoxy (45,8);
cprintf ("Subtotal        :");
gotoxy (45,9);
cprintf ("Total Tax       :");
gotoxy (43,10);
cprintf ("-----");
gotoxy (45,11);
cprintf ("Net Due         :");
}

```

```

/*-----
--
display_values_scr1_end ()
-----
--*/
display_values_scr1_end ()
{
    display_card_name_end ();
    display_card_number_end ();
    display_card_expr_end ();
    display_agreement_end ();
    display_phone_number_end ();
    display_batteries_end ();
    display_batteries_rented_end ();
    display_chargers_end ();
    display_chargers_rented_end ();
    display_discount_end ();
    display_rtb_charges_end ();
    display_days_charge_end ();
    display_totaltax_end ();
    display_other_end ();
    display_subtotal_end ();
}

```


ENDAGR.C

```

    display_netdue_end ();
}

/*-----
--
do_cancel_key
-----
--*/

do_cancel_key (int *done) {
    if (CARD_APPROVED) {
        errrtn ("Can't Cancel Now!");
    } else {
        error_wt = windowopen (&error_win);
        setttitle (error_wt, " F2 - CANCEL! ", CenterUpperTitle);
        gotoxy (5,2);
        if (yes_no ("Cancel Return, Are you sure (Y/N)?", FALSE)) {
            undo_return (); /* in realtime.c */
            *done = TRUE;
        }
        windowclose (error_wt);
        use (data_wt_end);
    }
}

/*-----
--
do_macro_key
-----
--*/

do_macro_key () {
    int stat;
    char errmessage[80];
    wintype wt;
    stat = w_is_next_end ();
    if (stat < CREDIT_STEP) {
        strcpy (errmessage, "Must Enter Chargers and Batteries
Returned");
        errrtn (errmessage);
    } else
    if (stat == CREDIT_STEP) {
        do_credit_end ();
    }
    if (CARD_APPROVED) {
        add_upd_agreemnt (2); /* 2 = ending agreement */
        strcpy (call_rec.agreemntno, agreemntrec.agreeno, 13);
        /* as flat file records */
        if (!PRINTED_CONTRACT) {
            print_contract (2, FALSE); /* final contract
printing == 2 */
            if (prt_error_number != 0) {
                strcpy (errmessage, prt_error_message);
                errrtn (errmessage);
            }
        }
    }
}

```


ENDAGR.C

```

Printer",CenterUpperTitle);
        wt = windowopen (&note_win);
        setttitle (wt,"Bypassing
printing the agreement ? (Y/N)",FALSE)) {
            PRINTED_CONTRACT = TRUE;
            update_tau_status (2,'5');
            w_log_end (PRINTING_STEP);
        }
        windowclose (wt);
    } else {
        w_log_end (PRINTING_STEP); /* log
successful print*/
        PRINTED_CONTRACT = TRUE;
    }
}
use (data_wt_end);
}

/*-----
do_print_key
-----*/
do_print_key () {
    int stat;
    wintype wt;
    char errmessage[80];
    stat = w_is_next_end ();
    if (stat >= PRINTING_STEP) {
        add_upd_agreemnt (2); /* 2 = ending agreement */
        strncpy (call_rec.agreemntno,agreemntrec.agreeno,13);
        /* as flat file records */
        print_contract (2,FALSE);
        if (prt_error_number != 0){
            strcpy (errmessage,prt_error_message);
            errrtn (errmessage);
            wt = windowopen (&note_win);
            setttitle (wt,"Bypassing Printer",CenterUpperTitle);
            beep ();
            gotoxy (1,2);
            if (yes_no ("Do you wish to bypass printing the
agreement ? (Y/N)",FALSE)) {
                PRINTED_CONTRACT = TRUE;
                update_tau_status (2,'5');
                w_log_end (PRINTING_STEP);
            }
            windowclose (wt);
        } else {
            w_log_end (PRINTING_STEP); /* log successful
print*/

```


ENDAGR.C

```

                                PRINTED_CONTRACT = TRUE;
                                }
                                } else {
                                    strcpy (errmsg, "Do Credit Authorization and Data Entry
Before Printing!");
                                    errrtn(errmessage);
                                }
                                use (data_wt_end);
                                }

/*-----
--
do_exit_key
-----
--*/
do_exit_key (int *done) {
    int stat;
    char errmsg[80];
        stat = w_is_next_end ();
        if (stat == EXIT_STEP) {
            update_tau_status (0, '0'); /* phone is in */
            add_upd_agreemnt (2); /* 2 = ending agreement */
            strncpy (call_rec.agreemntno, agreemntrec.agreemntno, 13);
            /* as flat file records */
            save_calls_as_flat_records ();
            system ("ccopyit agreemnt. ");
            system ("ccopyit phone. ");
            system ("ccopyit callrec.dat");
            *done = TRUE;
        } else {
            strcpy (errmsg, "Must Complete Credit Auth. and
Printing");
            errrtn (errmsg);
        }
    }

/*-----
--
do_detail_key
-----
--*/
do_detail_key () {
    show_detail ();
    use (data_wt_end);
    clrscr ();
    display_scr1_end();
    display_values_scr1_end();
}

/*-----
--
do_bypass_key
-----

```


ENDAGR.C

```

--*/
do_bypass_key () {
char temp[80],errmessage[80];
int temp_key,payment_type; /* 1 = cash, 2 = check, 3 = none */
wintype wt;
pick_list_type list;
wintype help_wt;

    if (!CARD_APPROVED) {
        add_to_pick_list (&list,"Cash Payment ",1);
        add_to_pick_list (&list,"Check Payment",2);
        add_to_pick_list (&list,"NO Payment  ",3);
        help_wt = help_window ("Select a payment type and press the
<ENTER> key");
        payment_type = pick_list (&list,3,"Payment Type");
        if ( payment_type == K_ESC ) { windowclose ( help_wt ); return
; }

        switch (payment_type) {
            case 1:
                strcpy (agreemntrec.approved,"CASH");
                break;
            case 2:
                strcpy (agreemntrec.approved,"CHECK");
                break;
            case 3:
                strcpy (agreemntrec.approved,"NONE");
                break;
        }
        use (help_wt);
        windowclose (help_wt);
        help_wt = help_window ("Press the F3 key when finished");
        manual_wt = windowopen (&manual_win);
        setttitle (manual_wt,"Non Credit Card Payment",CenterUpperTitle);
        gotoxy (5,1);
        cprintf ("Amount : ");
        gotoxy (5,2);
        cprintf ("Remark : ");
        null_field (temp,80);
        temp_key = K_ESC;
        use (manual_wt);
        global_amount_paid = 0.0;
        while (temp_key != K_F3) {
            null_field (temp,80);
            sprintf (temp,"%4.2f",global_amount_paid);
            temp_key = get_line (temp,5,1,7,manual_wt,"Amount : ");
            global_amount_paid = atof (temp);
            if (temp_key != K_F3) {
                null_field (temp,80);
                strcpy (temp,agreemntrec.remarks4);
                temp_key = get_line
(temp,5,2,30,manual_wt,"Remark : ");
                strcpy (agreemntrec.remarks4,temp);
            }
            if ( (global_amount_paid > agreemntrec.netdue) ) {

```


ENDAGR:C

```

        sprintf (temp,"Amount paid, %4.2f, is
greater than the total bill, %4.2f",global_amount_paid, agreemntrec.net
due);

        errrtn (temp);
        temp_key = K_ESC;
    } /* require something typed in remarks4 */
    if ( (global_amount_paid < 0.0) ) {
        errrtn ("Amount paid cannot be negative.");
        temp_key = K_ESC;
    }
    if ( (agreemntrec.remarks4[0] == ' ') ||
        (agreemntrec.remarks4[0] == '\0') ) temp_key =
K_ESC;
    }
    windowclose (manual_wt);
    windowclose (help_wt);
    CARD_APPROVED = TRUE; /* done and approved */
    w_log_end (CREDIT_STEP); /* log credit */
    if (!CREDIT_WIN_OPEN){
        credit_wt = windowopen (&credit_win);
        setttitle (credit_wt,"Credit Card
Authorization",CenterUpperTitle);
    }
    use (credit_wt);
    clrscr ();
    cprintf ("      Authorization Number : %s",agreemntrec.approved);
    use (data_wt_end);
} else {
    strcpy (errmessage,"Payment already completed!");
    errrtn (errmessage);
}
}

/*-----
--
get_data_end
-----
--*/
get_data_end ()
{
    int FIELD,done,key,stat,temp_key;
    char temp[20];
    wintype wt;

    display_scr1_end();
    display_values_scr1_end();
    FIELD = 1;
    done = FALSE;
    while (!done ){
        switch (FIELD) {
            case 1: key = get_chargers_end ();
                    w_log_end (CHARGERS_STEP);
                    break;
            case 2: key = get_batteries_end ();

```


ENDAGR.C

```

                                w_log_end (BATTERIES_STEP);
                                break;
                                case 3: key = get_discount_end ();
                                break;
                                }
                                derive_other ();
                                display_values_scr1_end();
                                if (key == K_F1) {
                                    help_list_end ();
                                    use (data_wt_end);
                                }
                                if (key == K_F10) {
                                    command_list_end ();
                                    use (data_wt_end);
                                }
                                if (UP_FIELD) {
                                    if (FIELD > 1){
                                        --FIELD;
                                    } else
                                        if (FIELD == 1) FIELD = 3;
                                }
                                if (DOWN_FIELD) {
                                    if (FIELD < 3) {
                                        ++FIELD;
                                    } else
                                        if (FIELD == 3) FIELD = 1;
                                }

                                if (key == K_F2) do_cancel_key (&done);

                                if (key == FORCED_EXIT) done = TRUE;

                                if (key == K_F3) do_macro_key ();

                                if (key == K_F4) {
                                    do_remarks ();
                                    use (data_wt_end);
                                }

                                if (key == K_F5) do_print_key ();

                                if (key == K_F6) do_exit_key (&done);

                                if (key == K_F7)
                                    if ( !CARD_APPROVED ) { do_detail_key ();
                                    } else errrtn ("No changes in the bill can be made
at this point!");

                                if (key == K_F8) do_bypass_key ();

                                if (key == K_F9) {
                                    w_next_end (&FIELD);
                                    use (data_wt_end);
                                }
                                /* in whatnext.c */

```


ENDAGR:C

```

    } /* while loop end */
}

/*-----
---
command_list_end: show command list
-----
--*/
command_list_end ()
{
char c;

    commands_wt = windowopen (&commands_win);
    setttitle (commands_wt," Commands List ",CenterUpperTitle);
    gotoxy (1,2);
    cprintf ("          F1  - Quick Step Help");
    gotoxy (1,3);
    cprintf ("          F2  - Cancel, 'Get Me Out Key'");
    gotoxy (1,4);
    cprintf ("          F3  - Finish Key");
    gotoxy (1,5);
    cprintf ("          F4  - Add Remarks To Contract");
    gotoxy (1,6);
    cprintf ("          F5  - Print Receipt");
    gotoxy (1,7);
    cprintf ("          F6  - Exit, 'I am all done!'");
    gotoxy (1,8);
    cprintf ("          F7  - Show Detailed Billing");
    gotoxy (1,9);
    cprintf ("          F8  - Non Credit Card Payment");
    gotoxy (1,10);
    cprintf ("          F9  - What Do I Do Next (?) Key");
    gotoxy (1,11);
    cprintf ("                      ESC - EXIT ");
    while ((c = getch ()) != K_ESC) ;
    windowclose (commands_wt);
}

/*-----
---
help_list_end: show command list
-----
--*/
help_list_end ()
{
wintype win;
char c;

    commands_wt = windowopen (&commands_win);
    setttitle (commands_wt," Quick Step Help ",CenterUpperTitle);
    gotoxy (1,1);
    cprintf ("                      STEP");
    gotoxy (1,2);
    cprintf ("          -----");
    gotoxy (1,3);

```


ENDAGR:C

```

cprintf ("      1 - Put Phone in CTI Box");
gotoxy (1,4);
cprintf ("      2 - Enter Batteries, Chargers Rtn");
gotoxy (1,5);
cprintf ("      3 - Press F3 To Finish");
gotoxy (1,6);
cprintf ("      6 - You're all done!");
gotoxy (1,10);
cprintf ("          ESC - EXIT ");
while ((c = getch ()) != K_ESC) ;
    windowclose (commands_wt);
}

/*-----
display_remarks: show remarks screen
-----*/
display_remarks ()
{
    gotoxy (42,2);
    textbackground (BLUE);
    cprintf ("ESC");
    textbackground (BLACK);
    cprintf (" - Exit");
    textcolor (WHITE);
    display_remarks1_end ();
    display_remarks2_end ();
    display_remarks3_end ();
}

/*-----
do_remarks: Allow Data Entry For Remarks
-----*/
do_remarks ()
{
    int FIELD, key, done;

    done = FALSE;
    FIELD = 1;
    remarks_wt = windowopen (&remarks_win);
    setttitle (remarks_wt, "Add Remarks To Contract", CenterUpperTitle);
    display_remarks ();
    while (!done) {
        switch (FIELD) {
            case 1: key = get_remarks1_end ();
                    break;
            case 2: key = get_remarks2_end ();
                    break;
            case 3: key = get_remarks3_end ();
                    break;
        }
        if (key == K_ESC)
            done = TRUE;
    }
}

```


ENDAGR.C

```
if (key == K_F2)
    done = TRUE;
if (key == K_F6)
    done = TRUE;
if (UP_FIELD) {
    if (FIELD > 1){
        --FIELD;
    } else
        if (FIELD == 1) FIELD = 3;
}
if (DOWN_FIELD) {
    if (FIELD < 3) {
        ++FIELD;
    } else
        if (FIELD == 3) FIELD = 1;
}
} /* end while */
windowclose (remarks_wt);
}
```


FCOPY.C

```

/****( fcopy.c )****
**/
/*
*/
/* PRO-C - Copyright (c) 1988 Vestronix Inc.
*/
/* 18 OCT 88
*/
/*
*/
/*****
**/
/* Procedure Name : FCOPY
*/
/*
*/
/* This routine is used to copy null terminated strings into a
*/
/* a field of a file record. The destination field is then padded out
*/
/* to a len of DLEN with nulls.
*/
/*
*/
/* Parameters:
*/
/*
*/
/* NAME TYPE DESCRIPTION
*/
/* ---- ----
*/
/* DEST STRING Destination field.
*/
/*
*/
/* SRC STRING Source string.
*/
/*
*/
/* DLEN INT Length of destination field
*/
/*****
**/
/* LINTLIBRARY */
#include <stdio.h>
#include <bench.h>

char *fcopy(dest, src, dlen)
char *dest;
char *src;
int dlen;
{
    char *p = dest;

```


FCOPY.C

```
if (src != NULL)
    while (*src != '\0' && dest < p + dlen)
        *dest++ = *src++;

while (dest < p + dlen)
    *dest++ = '\0';

return(dest);
}
```


FMTF.C

```

/****( fntf.c )****
**/
/*
*/
/* PRO-C - Copyright (c) 1988 Vestronix Inc.
*/
/* 18 OCT 88
*/
/*
*/
/*****
**/

#include <bench.h>
#include <ctype.h>

/* Function prototypes */
# ifdef ANSI
static char fmtwd(char *,char *);
static void reverse(char *);
# else
static char fmtwd();
static void reverse();
# endif

int negative = FALSE;
static int sign_done;

char *fmtflt(double, char *);

char *fmtflt(n, mask)
double n;
char *mask;
{
    return(fmtdbl((double)n, mask));
}

char *fmtdbl(num, msk)
double num;
char *msk;
{
    char c, *p, dc, *dp;
    char num1[81], *numr;
    static char mask1[41];
    char *maskr;
    int overflow = FALSE;

    /* break the mask into two parts, one for each side of the decimal
*/
    strcpy(mask1, msk);
    for(dp = mask1; (dc = *dp) != '\0' && dc != '.'; ++dp)
        ;

```


FMTF.C

```

if(dc != '\0')
    *dp++ = '\0';
maskr = dp;

/* find out how many decimal places were requested */
for(p = maskr; (c = *p) == '9' || c == 'z' || c == 'Z'; ++p)
    ;

/* under TC, the following are true: -0 != 0, -(-0) == -0 */
/* so, check for -0 here and set it to 0 to prevent trouble */
if(num == -0.0)
    num = 0.0;

/* convert number to non-negative and remember sign */
if(negative = (num < 0.0))
    num = -num;

/* let sprintf() do the work of actually converting it */
sprintf(num1, "%.*lf", (int)(p-maskr), num);

/* break the number into two parts, one for each side of the
decimal */
for(p = num1; (c = *p) != '\0' && c != '.'; ++p)
    ;
if(c != '\0')
    *p++ = '\0';
numr = p;

/* remove any trailing 0s for now, format '9' will restore them */
/*
* This bit is obsolete and doesn't work.
*/
for(p += strlen(p) - 1; *p == '0'; --p)
    ;
*++p = '\0';
*/

/* format each half separately, leaving the result in the mask
argument */
/* note that both parts are formatted from the decimal point
outwards */
sign_done = FALSE;
reverse(num1);
reverse(mask1);
overflow = (fmtwd(num1, mask1) != '\0');
reverse(mask1);

/* the part to the right of the decimal CANNOT overflow */
(void)fmtwd(numr, maskr);

/* put the decimal point back in and return the result */
if(dc != '\0')
    dp[-1] = '.';

```


280

Page 2

FMTF.C

```

    /* put in a sign if negative, there is none yet, and there is room
    */
    if(negative && !sign_done) {
        for(p = mask1; *p == ' '; ++p)
            ;
        if(p == mask1)
            overflow = TRUE;
        else
            *--p = '-';
    }

    /* set overflow indicator if overflow occurred */
    if(overflow)
        *mask1 = '?';

    return(mask1);
}

```

```

static char fmtwd(num, mask)
char *num, *mask;
{

```

```

    char mc, nc, *root;

    root = mask;
    while((mc = *mask) != '\0') {
        switch (mc) {
            case '9':
            case 'Z':
            case 'z':
                /* copy in digit if any left, otherwise copy '0'
                or ' ' */
                if((nc = *num) != '\0') {
                    ++num;
                    *mask = nc;
                }
                else
                    *mask = (mc == '9') ? '0' : ' ';
                break;
            case '+':
            case '-':
                /* treat as a sign only if this is the first one
                encountered */
                if (!sign_done)
                    /* next line allows non-sign + or - inside a num
                eg. III-III */
                    && mask[1] != 'Z' && mask[1] != 'z' && mask[1] !=

```


Page 3

FMTF.C

```

for(p = mask; --p >= root && *p == ' '; )
    *mask = ' ';
*++p = negative ? '-' : ((mc == '+') ? '+'
: ' ');
    }
    break;
case ',':
    /* don't want to delete comma if there is going to
be a zero */
    if(*num == '\0' && (mask[1] == 'z' || mask[1] ==
'Z'))
        *mask = ' ';
    break;
default:
    /* simply leave the character as is */
    break;
}
++mask;
}

/* return next character in number; this is used to detect
overflow */
return(*num);
}

```

```

/* not a very general function, but it's only used in this file */
static void reverse(str)
char *str;
{
    char tmp, *estr;
    for (estr = str + strlen(str) - 1; estr > str; estr--, str++) {
        tmp = *estr;
        *estr = *str;
        *str = tmp;
    }
}

```


GBASE.C

```

/*-----
---
MODULE : gbase.c      .... Phone Call Sequential data base.

Written By : Greg McGregor 1990

PURPOSE:
    It a sequential data base for storing phone calls.
    The neat thing about this is that every gbaserec is of different
size.
    So, all records stored on disk and in memory are of variable length.
    This saves having to allocate 250K for a record on disk for 1 call
versus
    1-2K for the record.  However, it is slow when a record is updated
or
    added.  It has to rebuild the entire database.  (A tradeoff)

REVISED:                What was revised?
GMM 7-30-1991           Nothing
-----
-*/

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include <fcntl.h>
#include <sys\stat.h>
#include <alloc.h>
#include <mem.h>
#include <string.h>
#include <\h2\malloc\galloc.h>

#define TRUE 1
#define FALSE 0

/*
 * record type
 */
typedef struct record_type_node {
    char tau_id[5];                /* tau id */
    char agreeno[20];              /* agreement number */
    char number[40];               /* allow for country */
    char start_time[10];           /* start time */
    char end_time[10];             /* end time */
    char date[10];                 /* date of call */
    float length;                  /* length of call minutes */
    float actual_secs;             /* actual call in seconds */
    float length_secs;             /* length in seconds */
    float total_cost;              /* total cost of call */
    float long_dist_cost;          /* long distance charges */
    float base_cost;              /* base charges */
    char flag;                     /* flags see flags */

```


GBASE.C

```
    struct record_type_node *next;  
} record_type;
```

```
/*  
 * typedef gbaserec KEY RECORD  
 */  
typedef struct grec {  
    int attached_records; /* # of attached records */  
    long size_of_rec; /* size of entire record block */  
    record_type *rec; /* LINKED List of records */  
    char agreemntno[20]; /* string */  
} gbaserec;
```

```
/*  
 * FLAGS Variable  
 */  
/*  
    bit 0 = ROAM ON;  
    bit 1 = Call with no connect to cell;  
    bit 2 = Out of memory  
    bit 3 = TBD  
    bit 4 = TBD  
 */
```

```
/*  
 * FUNCTIONS  
 */
```

```
record_type *g_get_call (gbaserec rec, int num);
```

```
/*  
 * GLOBALS  
 */  
int open_file = FALSE;  
int open_file_fd;  
int temp_file_fd;  
record_type *new_rec ();  
char fname[] = "rtb.lst";  
int file_flags;
```

```
gbaserec holding_rec, test;
```

```
/*  
 * Last Includes  
 */  
#include <time.h>  
#include <\h2\hdr\windows.h>  
#include <\h2\hdr\extnvar.h>
```


GBASE.C

```

/*
 *
main ()
{
int i;
record_type *s;

    system ("del temp.xxx");
    system ("del rtb.lst");
    clrscr ();
    if ( (i = g_open ("rtb.lst",O_RDWR,&holding_rec)) == -1) printf
("\nerror");
    s = new_rec ();
    strcpy (s->number,"415-838-2400");
    assoc_rec (&holding_rec,s);
    s = new_rec ();
    strcpy (s->number,"415-838-2401");
    assoc_rec (&holding_rec,s);
    s = new_rec ();
    strcpy (s->number,"415-838-2481");
    assoc_rec (&holding_rec,s);
    strcpy (holding_rec.agreemntno,"SHIT1");
    g_put (holding_rec);
    strcpy (holding_rec.agreemntno,"SHIT2");
    g_put (holding_rec);
    g_close (i);
    if ( (i = g_open ("rtb.lst",O_RDWR,&holding_rec)) == -1) printf
("\nerror");
    strcpy (test.agreemntno,"SHIT2");
    g_get (&test);
    s = g_get_call (test,2);
    strcpy (s->number,"111-111-1111");
    g_close (i);
}
*/

/*-----
new_rec
-----*/
record_type *new_rec()
{
    return (record_type *)g_malloc (sizeof (record_type));
}

/*-----
garbage_collect : free up all call records in memory
-----*/
garbage_collect (gbaserec *call_rec) /* pass by reference MUST */
{

```


GBASE.C

```

int i,j; /* no calls made */
record_type *a_call;

    i = call_rec->attached_records;
    for (j = i; j >= 1; j--) {
        a_call = g_get_call (*call_rec,j); /* get a call from
memory */
        g_free (a_call); /* free it from memory */
    }
    /* reset call_rec info */
    call_rec->attached_records = 0;
    call_rec->rec = NULL; /* null out pointer to calls */
    call_rec->agreemntno[0] = '\0'; /* null agreemntno */
    /* gbase garbage_collecting done*/
}

/*-----
g_open:  flags - O_RDONLY O_WRONLY O_RDWR
-----*/
/
int g_open (char *name, int flags, gbaserec *r)
{
    int fd;
    int i;
    if (open_file) return -1;
    if ( (fd = open (name,flags|O_BINARY|O_CREAT,S_IREAD|S_IWRITE)) ==
-1)
        return -1; /* couldn't open or create the file */
    r->attached_records = 0;
    r->size_of_rec = sizeof (gbaserec);
    open_file = TRUE;
    open_file_fd = fd;
    strcpy (fname,name);
    file_flags = flags;
    return fd;
}

/*-----
g_close
-----*/
int g_close (int fd)
{
    int i,j;
        j = close (fd);
        if (j == -1) return -1;
        open_file = FALSE;
        return 0;
}

/*-----
assoc_rec: attach record onto gbaserec
-----*/

```


GBASE.C

```

*/
int assoc_rec (gbaserec *r, record_type *a)
{
    int i;
    record_type *x;
    i = r->attached_records;
    if (i == 0) {
        r->rec = a;
        ++r->attached_records;
        r->size_of_rec = r->size_of_rec + sizeof (*a);
        return TRUE;
    }
    /* find place to put record */
    x = r->rec;
    --i;
    while (i > 0) {
        x = x->next;
        --i;
    }
    x->next = a;
    a->next = NULL;
    ++r->attached_records;
    r->size_of_rec = r->size_of_rec + sizeof (*a);
}

/*-----
---
g_get: get record from disk
        need in param 'r'  tag,?_key
-----
--*/
int g_get (gbaserec *r)
{
    long offset;
    gbaserec temp;
    int done,stat;
    int num_assoc_recs;
    record_type *rt;

    done = FALSE;
    lseek (open_file_fd,0L,SEEK_SET);      /* go to beginning of file */
    do {
        stat = read (open_file_fd,&temp,sizeof (gbaserec));
        if (stat == -1) /* ERROR */
            return -1;
        if (stat == 0) return -1; /* NOT FOUND*/
        if (strcmp (temp.agreemntno,r->agreemntno) == 0) done =
TRUE;
        if (!done) { /* skip associated records */
            offset = temp.size_of_rec - sizeof (gbaserec);
            lseek (open_file_fd,offset,SEEK_CUR);
        }
    } while (!done);
}

```


GBASE.C

```

/* still have to read in associated records, if there are some */
num_assoc_recs = temp.attached_records;

if (temp.attached_records == 0) return 0; /* no attached recs */

*r = temp;
r->attached_records = 0;
r->size_of_rec = sizeof (gbaserec);
while (num_assoc_recs > 0) {
    rt = new_rec ();
    stat = read (open_file_fd,rt,sizeof (record_type));
    if (stat == -1) /* error */
        return -1;
    if (stat == 0) return 0; /* EOF */
    assoc_rec (r,rt); /* attach call to record in memory */
    --num_assoc_recs;
}
return 1;
}

```

```

/*-----
---
g_get_next: get next record from disk from current file pointer
-----*/
int g_get_next (gbaserec *r)
{
    int num_assoc_recs,stat;
    record_type *rt;

    stat = read (open_file_fd,r,sizeof (gbaserec));
    if (stat == -1) /* ERROR */
        return -1;
    if (stat == 0) /* EOF */
        return 0;
    /* still have to read in associated records, if there are
some */
    num_assoc_recs = r->attached_records;

    if (r->attached_records == 0) return 0; /* no attached recs */

    r->attached_records = 0;
    r->size_of_rec = sizeof (gbaserec);
    while (num_assoc_recs > 0) {
        rt = new_rec ();
        if ( read (open_file_fd,rt,sizeof (record_type)) == -1)
            return -1;
        assoc_rec (r,rt); /* attach call to record in memory */
        --num_assoc_recs;
    }
    return 1;
}

```


GBASE.C

```

/*-----
g_write_temp: put record to disk in temp file, a helper to g_update
-----*/
/
int g_write_temp (gbaserec r)
{
    int num_recs;
    record_type *rt;

    num_recs = r.attached_records;
    if (write (temp_file_fd,&r,sizeof (gbaserec)) == -1)
        return -1;
    rt = r.rec;
    while (num_recs > 0) {
        if (write (temp_file_fd,rt,sizeof (record_type)) == -1)
            return -1;
        rt = rt->next;
        --num_recs;
    }
    return 1;
}

/*-----
--
g_update : update a record to disk
-----
--*/
int g_update (gbaserec r)
{
    long offset;
    gbaserec temp;
    int done,i,updated;
    char sys[80];

    done = FALSE;
    updated = FALSE;
    if ( (temp_file_fd = open
("temp.xxx",O_WRONLY|O_BINARY|O_CREAT|O_TRUNC,S_IWRITE)) == -1)
        return -1;    /* couldn't open or create the file */

    lseek (open_file_fd,0L,SEEK_SET);    /* go to beginning of file */

    do {
        if (g_get_next (&temp) == 0)
            done = TRUE;
        if( (strcmp (temp.agreemntno,r.agreemntno) != 0) &&
(!done) ){
            g_write_temp (temp);
        } else {
            if (!updated) {
                g_write_temp (r);
                updated = TRUE;
            }
        }
    }
}

```


GBASE.C

```

    } while (!done);

    close (temp_file_fd);
    g_close (open_file_fd);

    strcpy (sys, "copy temp.xxx ");
    strcat (sys, fname);
    strcat (sys, " >out");
    system (sys);

    if ( (i = g_open (fname, O_RDWR, &temp)) == -1) {
        printf ("\ng_put: error updating call record data base!");
        return -1;
    }
    open_file_fd = i;

    if (g_get (&r) == -1) {
        printf ("\ng_put: error updating call record data base!");
        return -1;
    }
    return 1;
}

```

```

/*-----
--
g_write : write a gbaserec to disk
-----
--*/
int g_write (gbaserec r)
{
    int num_recs;
    record_type *rt;

    num_recs = r.attached_records;
    if (write (open_file_fd, &r, sizeof (gbaserec)) == -1)
        return -1;
    rt = r.rec;
    while (num_recs > 0) {
        if (write (open_file_fd, rt, sizeof (record_type)) == -1)
            return -1;
        rt = rt->next;
        --num_recs;
    }
    return 1;
}

```

```

/*-----
--
g_put : put a record to disk update if record already exists
-----
--*/
int g_put (gbaserec r)

```


GBASE.C

```

{
long offset;
gbaserec temp;
int done,i,updated;
char sys[80];

    if (g_exists (r) ) {
        return g_update (r);
    }
    lseek (open_file_fd,0L,SEEK_END);

    return g_write (r);
}

/*-----
g_exists : does a record exist
-----*/
/
int g_exists (gbaserec rec) {
gbaserec *temp;

    lseek (open_file_fd,0L,SEEK_SET);      /* go to beginning of file */
    while (g_get_next (temp) == 1) {
        if (strncmp (temp->agreemntno,rec.agreemntno,13) == 0)
            return TRUE;
    }
    return FALSE;
    lseek (open_file_fd,0L,SEEK_SET);      /* go to beginning of file */
}

/*-----
g_get_call:  get a call given a call number identifier
              whatever you change in the return record here
              gets changed
              int the rec you pass not just in what you get
              returned
-----*/
/
record_type *g_get_call (gbaserec rec, int num)
{
int i;
record_type *temp;

    i = rec.attached_records;
    if ( (i < num) || (num <= 0) ) return NULL;
        /* record doesn't exist  so return NULL REC*/
    i = 1;
    temp = rec.rec;
    if (num == 1) return temp;
    do {
        temp = temp->next;
        ++i;
    } while (i != num);
}

```


GBASE.C

```

    return temp;    /* got and return */
}

/*-----
save_call_records
-----*/

/
save_call_records () {
    gbaserec x;
    int fd;
    int stat;
    windef call_win = {10,8,70,12,White,Red,FALSE,FALSE,FALSE,TRUE,SINGLEF
RAME,
                                White,Red};
    wintype call_wt;

    call_wt = windowopen (&call_win);
    setttitle (call_wt,"Call Accounting System..",CenterUpperTitle);
    gotoxy (1,2);
    cprintf ("                Updating Call Accounting System...");
    fd = g_open ("callrec.dat",O_RDWR,&x);
    stat = g_put (call_rec);
    if (stat != 1) {
        errrtn ("Error In Updating Call Accounting System!");
    }
    g_close (fd);
    windowclose (call_wt);
}

/*-----
g_save_as_flat (call_rec);
-----*/

/
g_save_as_flat (gbaserec call_rec)
{
    int fd;
    int i,j,stat;
    record_type *call;
    fd = open
("callrec.dat",O_RDWR|O_BINARY|O_CREAT|O_APPEND,S_IWRITE|S_IREAD);
    if (fd <= 0) {
        printf ("\nERROR (g_save_as_flat):  File Open Error
CALLREC.DAT");
        printf ("\n  Call Telemac Cellular Corporation
(800)-236-2356");
        exit (1);
    }
    i = call_rec.attached_records;
    for (j=1;j<=i;j++) {
        call = g_get call (call_rec,j);

```


GBASE.C

```

Error CALLREC.DAT");
(800)-235-2356");
    printf ("\nERROR (g_save_as_flat):  File Write
    printf ("\n Call Telemac Cellular Corporation
    exit (1);
}
}
close (fd);
return TRUE;
}

```

```

/*-----
save_calls_as_flat_records
-----*/
/
save_calls_as_flat_records () {
gbaserec x;
int fd;
int stat;
windef call_win = {10,8,70,12,White,Red,FALSE,FALSE,FALSE,TRUE,SINGLEF
RAME,
                                White,Red};
wintype call_wt;

call_wt = windowopen (&call_win);
settitle (call_wt,"Call Accounting System..",CenterUpperTitle);
gotoxy (1,2);
cprintf ("                Updating Call Accounting System...");
fd = g_open ("callrec.dat",O_RDWR,&x);
stat = g_save_as_flat (call_rec);
if (stat != 1) {
    errrtn ("Error In Updating Call Accounting System!");
}
windowclose (call_wt);
}

```


GETLINE.C

```

/*-----
--
getline.c
    getting lines from console

Written By : Greg McGregor

REVISED:                What was revised?
GMM 7-30-1991            Nothing
-----
-*/

#include <bios.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <windows.h>
#include <misc.h>
#include <gkeys.h>
#include <time.h>
#include <whatopen.h>
#include <proc.io>
#include <bench.h>
#include <gbase.h>
#include <extnvar.h>
#include <agriio.h>
#include <agreev3.h>
#include <cardrdr.h>

extern int UP_FIELD;
extern int DOWN_FIELD;
extern int FIELD;
extern W_PRINTED;

/*-----
// Function Name -> capAdjustNoleft
// Parameters:
// Function: capitalize but don't left adjust VS capAdjust function
// Returns:
// Written By : Greg McGregor
//
-----*/
void capAdjustNoleft (char *s, int max) {
    int i;
    i = 0;
    while ( (s[i]) && (i <= max) ) {
        if (islower (s[i])){
            s[i] = toupper (s[i]);
        }
        ++i;
    }
}

```


GETLINE.C

```

/*-----
capAdjust: left justify and capitalize all alpha null ended fields
-----*/
void capAdjust (char *s,int max)
{
    int i,len,j,t,get_out;
    get_out = FALSE;
    len = strlen (s);
    i = j = 0;
    while ( (s[i++] == ' ') && (!get_out) )
        if (i >= max) get_out = TRUE;    /* nothing in field, get o
ut*/
    --i;
    while (s[j++] = s[i++]) ;
    i = 0;
    while ( (s[i]) && (i <= max) ) {
        if (islower (s[i])){
            s[i] = toupper (s[i]);
        }
        ++i;
    }
}

/*-----
debug_printX  print raw data
-----*/
debug_printX (unsigned char *s,int len)
{
    int i;
    for (i=0;i<len;i++) {
        cprintf ("%c",s[i]);
    }
    cprintf ("/");
    for (i=0;i<len;i++) {
        cprintf ("%X ",s[i]);
    }
}

/*-----
Xcmp : byte compare, compares two objects
-----*/
int Xcmp (char *s,char *s1,int len)
{
    int i,j;
    i = 0;
    for (j = 0;j<len;j++) {
        if (s[j] < s1[j]) /* less than */
            return -1;
    }
}

```


GETLINE.C

```

        if (s[j] > s1[j]) /* greater than */
            return 1;
    }
    return 0; /* equal */
}

/*-----
remove_char (char *s,int index)
-----*/
void remove_char (char *s,int index)
{
    while (s[index]) {
        s[index] = s[index+1];
        ++index;
    }
    s[index] = '\0';
}

/*-----
shorten_blanks:  Take out any reoccurring Blanks
-----*/
shorten_blanks (char *s)
{
    int i,j;
    i = 0; j = 1;
    while (s[j]) {
        if ((s[i] == ' ') && (s[j] == ' ')) {
            remove_char (s,i);
        } else {
            ++j;
            ++i;
        }
    }
    s[++i] = '\0';
}

/*-----
centerPrint :
-----*/
centerPrint (int l,char s[])
{
    int len,i;
    len = strlen (s);
    len = len /2;
    l = l /2;
    len = l - len;
    for (i=1;i<=len;i++)
        cprintf (" ");
    cprintf ("%s",s);
}

/*-----

```


GETLINE.C

```
do_nothing:
-----*/
do_nothing()
{
}

/*-----
is_field_empty() : is field empty or does it have input
-----*/
is_field_empty(char s[])
{
    if ( (s[0] == '\0') || (s[0] == ' ') ) {
        return TRUE; /* since all fields are left justified */
    } else return FALSE;
}

/*-----
is_extended_key : is key pressed a extened key,if so store key in key
-----*/
/
int is_extended_key (char c,char *key)
{
    char ch;
    if (c == 0){
        ch = getch();
        *key = ch;
        return TRUE;
    }
    return FALSE;
}

/*-----
getchb() : getch with bios. Not on int 21 line DONT' GET DOS REENTRY
-----*/
int getchb ()
{
    return getch ();    /* TEMP */
}

/*-----
is_null_ended: is field null ended
-----*/
int is_null_ended (char *s,int max)
{
    int i,null_ended;
    null_ended = FALSE;
    for (i=0;i<max;i++)
        if (s[i] == '\0')
            null_ended = TRUE;
    return (null_ended);
}
```


GETLINE.C

```

/*-----
yes_no : true if yes false if no on yes/no question
-----*/
int yes_no (char *s,int d)
{
char ch;
int key;
    cprintf ("%s",s);
    if (d){
        cprintf (" [Y]");
        key = TRUE;
    } else {
        cprintf (" [N]");
        key = FALSE;
    }
    do {
        ch = getch();
    } while ( (ch != 'Y') && (ch != 'y') && (ch != 'n') && (ch != '
N')
                && (ch != K_RETURN) );
    if ( (ch == 'y') || (ch == 'Y') )
        key = TRUE;
    if ( (ch == 'n') || (ch == 'N') )
        key = FALSE;
    if ( ch == K_RETURN) key = d;
    return key;
}

```

```

cprintfN (char s[],int n)
{
int i;
    i = 0;
    while ( (i<n) && (s[i]) ){
        cprintf ("%c",s[i]);
        ++i;
    }
}

```

```

/*-----
print_mask:
-----*
/
print_mask (char s[],char mask[],int len)
{
int i;
    i = 0;
    while (i <= len){
        if (mask[i] == ' ') {
            cprintf ("%c",s[i]);
            i++;
        } else {
            cprintf ("%c",mask[i]);

```


GETLINE.C

```

    }
}

/*-----
move_mask
-----*/
move_mask (char *s,char *mask,int max) {
int i = 0;
    for (i=0;i<max;i++)
        if (mask[i] != ' ') s[i] = mask[i];
}

/*-----
is_char_in_mask
-----*/
int is_char_in_mask (char ch,char *mask) {
char *item;
    item = mask;
    while (*item) {
        if ( (ch == *item) && (ch != ' ') )
            return ( TRUE );
        ++item;
    }
    return ( FALSE );
}

/*-----
find_position_after_mask
-----*/
int find_position_after_mask (int pos,char *mask) {
char *item;
int i,save;
    item = mask;
    save = pos;
    for (i=0;i<pos;i++) /* move to pos */
        ++item;
    while (*item == ' ') {
        ++item;
        ++pos;
    }
    if (!is_char_in_mask (*item,mask)) return ( save );
    ++pos; /* move past mask char */
    return pos;
}

/*-----
get_line_mask: get a string from console, allow for time slicing
NOTE!: First and Last CHAR cannot be masked
-----*/

```


GETLINE.C

```

-----*
/
int get_line_mask (s,x,y,max,win,prompt,mask)
char *s;
int x,y,max;
wintype win;
char prompt[];
char mask[];
{
int i,pos,done = FALSE;
int s_pos;
char c,key;
time_t tt,start,now;
int kick_out;

    _setcursortype (_NORMALCURSOR);
    s_pos = 0;
    pos = 0;
    c = ' ';
    if (PRINTED_CONTRACT) {
        kick_out = 30 * CLK_TCK;
    } else kick_out = 900 * CLK_TCK;
    /* cursoron (); */
    UP_FIELD = FALSE;
    DOWN_FIELD = FALSE;
    if (s[0] == '\0') {
        /* Blank field if necessary */
        for (i=0;i<max;i++)
            s[i] = ' ';
        s[i]='\0';
    }

    gotoxy (strlen(prompt)+x,y);
    textbackground (Blue);
    move_mask (s,mask,max);
    if (!is_null_ended (s,max)) {
        cprintfN (s,max);
    } else cprintf ("%s",s);

    if (strlen (s) < max)
        for (i=1;i<= (max - strlen(s));i++)
            cprintf (" ");

    gotoxy (x,y);
    textbackground (Black);
    cprintf ("%s",prompt);
    textbackground (Blue);
    while (!done) { /* return key */
        start = clock ();
        if (pos > max) {
            s_pos = pos = 0;
            gotoxy (x+strlen (prompt),y);
        }
        while (!kbhit() ) {
            now = clock ();
            if ( (now - start) > kick_out) {

```


GETLINE.C

```

if (kick_out == 30*CLK_TCK){
    if (forced_exit ())
        return K_F6; /* save
agreemnt and exit */
} else
if (forced_exit ())
    return FORCED_EXIT;
start = clock (); /* makes it to
here start over*/
}
c = getch();
if (!is_extended_key (c,&key)) {
    if ((pos >= max) && (c != BACKSPACE)
        && (c != ENTER)) { /* too many chars */
        do_nothing();
    } else {
        if (c == BACKSPACE) {
            if (pos >= 1) {
                --pos;
                s_pos = pos;
                while (mask[pos] != ' ')
                    --pos;
                s[pos] = ' ';
                gotoxy (wherex() -(s_pos -
pos),wherey());

                cprintf ("%c",BACKSPACE);
                cprintf (" ");
                cprintf ("%c",BACKSPACE);
            } else {
                pos = max;
                gotoxy
(x+max+strlen(prompt),y);
            }
        } else {
            if (c == K_ESC) {
                done = TRUE;
                key = K_ESC;
            } else
            if (c == K_TAB) {
                DOWN_FIELD = FALSE;
                UP_FIELD = TRUE;
                done = TRUE;
            } else
            if (c == ENTER) {
                DOWN_FIELD = TRUE;
                UP_FIELD = FALSE;
                done = TRUE;
            } else { /*
regular char */
/* if they press a
mask char, jump past next */
/* mask */
if (is_char_in_mask

```


GETLINE.C

```

(c,mask) ) {
    find_position_after_mask (pos,mask);
    + (pos - s_pos), wherey());

    ("%c",c);

    (mask[pos] != ' ') && (pos <= max) )

    (wherex() + (pos - s_pos),wherey());
}

}
} else {
    if (key == LEFT) {
        if (pos >= 1) {
            --pos;
            s_pos = pos;
            while (mask[pos] != ' ')
                --pos;
            gotoxy (wherex() - ((s_pos - pos) + 1),wherey());
        } else {
            pos = max;
            gotoxy (x+max+strlen(prompt),y);
        }
    } else
    if (key == RIGHT) {
        if (pos < max) {
            ++pos;
            s_pos = pos;
            while (mask[pos] != ' ')
                ++pos;
            gotoxy (wherex() + ((pos - s_pos) + 1),wherey());
        } else {
            pos = 0;
            gotoxy (x+strlen(prompt),y);
        }
    } else
    if (key == UP ) {
        UP_FIELD = TRUE;
        DOWN_FIELD = FALSE;
        done = TRUE;
    } else
        if (key == DOWN) {
            DOWN_FIELD = TRUE;
            UP_FIELD = FALSE;

```


GETLINE.C

```

        done = TRUE;
    } else {
        done = TRUE;
        textcolor (White);
        textbackground (Black);
        gotoxy (x+strlen(prompt),y);
        for (i=1;i<=max;i++)
            cprintf (" ");
        gotoxy (x+strlen(prompt),y);
        capAdjustNoleft (s,max);
        s[max] = '\0';
        cprintf ("%s",s);
        return (key);
    }
}

textcolor (White);
textbackground (Black);
gotoxy (x+strlen(prompt),y);
for (i=1;i<=max;i++)
    cprintf (" ");
gotoxy (x+strlen(prompt),y);
capAdjustNoleft (s,max);
s[max] = '\0';
cprintf ("%s",s);
return (key);
}

/*-----
format_string
-----*/
format_string (char *s,char *f)
{
    int i,j,len;
    j = 0;
}

/*-----
get_line: get a string from console, allow for time slicing
-----*/
/
int get_line (char *s,int x,int y,int max, wintype win,char prompt[])
{
    int i,pos,done = FALSE;
    char c,key;
    time_t start,now;
    int kick_out;

    _setcursortype (_NORMALCURSOR);
    pos = 0;
    c = ' ';
    UP_FIELD = FALSE;

```


GETLINE.C

```

        DOWN_FIELD = FALSE;
if (s[0] == '\0') { /* Blank field if necessary */
    for (i=0;i<max;i++)
        s[i] = ' ';
    s[i]='\0';
}

gotoxy (strlen(prompt)+x,y);
textbackground (Blue);
if (!is_null_ended (s,max)) {
    cprintfN (s,max);
} else cprintf ("%s",s);
if (strlen (s) < max)
    for (i=1;i<= (max - strlen(s));i++)
        cprintf (" ");
gotoxy (x,y);
textbackground (Black);
cprintf ("%s",prompt);
textbackground (Blue);
if (PRINTED_CONTRACT) {
    kick_out = 30 * CLK_TCK;
} else kick_out = 900 *CLK_TCK;

while (!done) { /* return key */
    start = clock ();
    if (pos > max) {
        pos = 0;
        gotoxy (x+strlen (prompt),y);
    }
    while (!kbhit() ) { /* poke at clock, hope date
will change*/
        /* if left
here over midnight */

        now = clock ();
        if ( (now - start) > kick_out) {
            if (kick_out == 30*CLK_TCK){
                if (forced_exit ())
                    return
            } else
                if (forced_exit ())
                    return

            start = clock (); /*
makes it to here start over*/
        }
    }
    c = getch();
    if (c == '%') {
        do_card_reader ();
        UP_FIELD = FALSE;
        DOWN_FIELD = FALSE;
        return K_CARD_READER;
    }
}

```


GETLINE.C

```

        if (!is_extended_key (c,&key)) {
if ((pos >= max) && (c != BACKSPACE)
    && (c != ENTER)) { /* too many chars
*/
        do_nothing();
    } else {
        if (c == BACKSPACE) {
            if (pos >= 1) {
                pos = pos - 1;
                s[pos] = ' ';

                cprintf
("%c", BACKSPACE);

                cprintf (" ");
                cprintf
("%c", BACKSPACE);

            } else {
                pos = max;
                gotoxy (x+max+strlen(prompt),y);
            }
        } else {
            if (c == K_ESC) {

                done = TRUE;
                key = K_ESC;
            } else
            if (c == K_TAB) {
                DOWN_FIELD = FALSE;
                UP_FIELD = TRUE;
                key = K_TAB;
                done = TRUE;
            } else
            if (c == ENTER) {
                key = K_RETURN;

                DOWN_FIELD = TRUE;
                UP_FIELD = FALSE;
                done = TRUE;

            } else {

                s[pos] = c;
                pos = pos
+ 1;

                cprintf
("%c",c);

            }
        }
    } else {
        if (key == LEFT) {
            if (pos >= 1) {
                pos = pos - 1;
                gotoxy (wherex()-1,wherey());
            } else {
                pos = max;
                gotoxy (x+max+strlen(prompt),y);
            }
        }
    }
}

```


GETLINE.C

```

    } else
    if (key == RIGHT) {
        if (pos < max) {
            pos = pos + 1;
            gotoxy (wherex()+1,wherey());
        } else {
            pos = 0;
            gotoxy (x+strlen(prompt),y);
        }
    } else
    if (key == UP ) {
        UP_FIELD = TRUE;
        DOWN_FIELD = FALSE;
        done = TRUE;
    } else
    if (key == DOWN) {
        DOWN_FIELD = TRUE;
        UP_FIELD = FALSE;
        done = TRUE;
    } else {
        done = TRUE;
        textcolor (White);
        textbackground (Black);
        gotoxy (x+strlen(prompt),y);
        for (i=1;i<=max;i++)
            cprintf (" ");
        gotoxy (x+strlen(prompt),y);
        capAdjust (s,max);
        s[max] = '\0';
        cprintf ("%s",s);
        return (key);
    }
}

textcolor (White);
textbackground (Black);
gotoxy (x+strlen(prompt),y);
for (i=1;i<=max;i++)
    cprintf (" ");
gotoxy (x+strlen(prompt),y);
capAdjust (s,max);
s[max] = '\0';
cprintf ("%s",s);
return (key);
}

/*-----
do_card_reader : make a call to card reader module
-----*/
do_card_reader () {
    int stat;
    ungetch ('%'); /* put back the % , card_reader mod needs */
    stat = read_in_card (agreemntrec.creditno,

```


GETLINE.C

```

                                                                    agreemntrec.custname,
                                                                    agreemntrec.expiredate,
                                                                    agreemntrec.creditttype);
    if (!stat) {
        errrtn ("Card Reading Interrupted By USER!");
    }
}

/*-----
forced_exit : exit and save contract
-----*/
int forced_exit ()
{
    int i;
    int stop;
    wintype note_wt;
    windef error_win =
    {10,10,70,15,White,Red,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
        White,Red};

    stop = FALSE;
    note_wt = windowopen (&error_win);
    setttitle (note_wt,"* Returning To Main Menu *",CenterUpperTitle );
    gotoxy (1,3);
    centerPrint (60,"Press <SPACE BAR> to Stop!");
    i = 30;
    while ( (i >= 0) && (!stop) ) {
        sound (1000);
        delay (20);
        nosound ();
        delay (1000);
        if (kbhit ()) {
            getch ();
            stop = TRUE;
        }
        --i;
    }
    windowclose (note_wt);
    if (!stop) {
        return TRUE;
    } else return FALSE;
}
```


GSTRING.C

```

/*****

```

```

7/21/89  Greg McGregor

```

```

REVISED:          What was revised?

```

```

GMM 7-30-1991      Nothing

```

```

*****/

```

```

#include <alloc.h>

```

```

/*-----
strsize  :  size a string
-----*/

```

```

char *strsize (int x)

```

```

{
    char *p;
    p = (char *)malloc (x + 1);
    return p;
}

```

```

/*-----
str_image : malloc room for a string
-----*/

```

```

char *str_image (char *s)

```

```

{
    char *p;
    p = (char *) malloc (strlen (s) + 1);
    if (p != NULL)
        strcpy (p, s);
    return p;
}

```

```

/*-----
strmid   : proc link - version 1 to version 2.0 compatible link
-----*/

```

```

char *strmid (char *dest, char *source, int start, int len)

```

```

{
    int i;
    for (i=0; i<len; i++) {
        dest[i] = source[start+i];
    }
    dest[i] = '\0';
}

```

```

/*-----
null    : null out a string
-----*/

```

```

null (char *s)

```

```

{
    s[0] = '\0';
}

```

```

/*-----
null_field
-----*/

```


GSTRING.C

```

null_field (char *s,int l)
{
    int i;
        for (i=0;i<l;i++)
            s[i] = '\0';
}

```

```

/*****

```

```

* moveX (x,y,X)
* copy y to x only X char's and add NO NULL
*****/

```

```

int moveX (x,y,n)
char *x,*y;
int n;
{
    int i;
        for (i=0;i<n;i++)
            x[i] = y[i];
}

```

```

/*****

```

```

* strcpyX (x,y,X)
* params : two pointers char * or char[] but must be same
* returns : none
* copies X number of char's from y to x
*****/

```

```

strcpyX (to,from,X)
char *from,*to;
int X;
{
    int add = 0;
    int add1 = 0;
        while ((to[add] = from[add1]) && (add1 != X-1)) {
            ++add;
            ++add1;
        }
        to[++add] = '\0';
}

```

```

/*****

```

```

* strNUMcat
* params : char s[], and int t
* returns : none
* function: concatenates t to s
*****/

```


GSTRING.C

```
strNUMcat (s,t)
char s[];
int t;
{
    int i;
        i = 0;
        while (s[i] != '\0') ++i;    /* find end of string */
        s[i] = t;
        s[i+1] = '\0';    /* have to add null to end because t has no nu
}}*/
}
```

/* cats a char to end of string */

```
strCHcat (s,t)
char *s;
char t;
{
    int i;
        i = 0;
        while (s[i] != '\0') ++i;
        s[i] = t;
        s[i+1] = '\0';
}
```


LOSTDAM.C

```
/*-----  
-----  
MODULE: lostdam.c V1.00  
Lost and Damaged phone return process  
  
CREATED:  
GMM 8-11-1991  
  
REVISED:  
-----  
-*/  
  
#include <stdio.h>  
#include <ctype.h>  
#include <bench.h>  
#include <proc.io>  
#include <sys\stat.h>  
#include <windows.h>  
#include <gkeys.h>  
#include <misc.h>  
  
#include <agreev3.h> /* struct formats */  
#include <control.h>  
#include <phone.h>  
#include <raperson.h>  
#include <agrioh.h>  
#include <gbase.h>  
#include <time.h>  
#include <extnvar.h>  
#include <getline.h>  
#include <taustat.h>  
#include <extscrns.h>  
  
windef big_note_win  
={5,11,75,20,White,Black,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,  
  White,Black};  
  
wintype big_note_wt;  
  
/*-----  
prompt_for_phone_number  
-----*/  
prompt_for_phone_number ()  
{  
  wintype win;  
  char number [20];  
  int key;  
  win = windowopen (&big_note_win);  
  setttitle (win,"Retrieving Contract",CenterUpperTitle);
```


LOSTDAM.C

```

strcpy (number, "
");
gotoxy (5,3);
cprintf ("Please enter the phone number of the Cellular Phone");
gotoxy (5,4);
cprintf ("so that the contract may be closed.");
key = get_line_mask (number,5,6,12,win,"Phone Number -> ", "
");
selectinx9 (fd_agreemnt,3);
moveX (agreemntrec.curphoneno,number,12);
windowclose (win);
}

/*-----
pull_phonelist
-----*/
int pull_phonelist () {
int iostat;
struct phone_def temp_phonerec;

iostat = selectinx9 (fd_phone,1);
strncpy (phonerec.curphoneno,agreemntrec.curphoneno,12);
iostat = exactkey9 (fd_phone,&phonerec);
if (iostat < 0) {
return FALSE;
} else return TRUE;
}

/*-----
pull_contract_by_phone
-----*/
int pull_contract_by_phone () {
int iostat;
int found = FALSE;
struct agreemnt_def temp_agreemnt;
char agreeno_save[20];

iostat = 0;
selectinx9 (fd_agreemnt,3);
iostat = reset_file9 (fd_agreemnt, &temp_agreemnt);
iostat = exactkey9(fd_agreemnt, &agreemntrec);
if (iostat < 0) {
errrtn ("Can't Find Agreement!");
} else found = TRUE;
do{
moveX(agreeno_save,agreemntrec.agreeno,12);
iostat = nextkey9(fd_agreemnt, &agreemntrec);
if (iostat == 0)
moveX(agreeno_save,agreemntrec.agreeno,12);
} while (iostat == 0);
selectinx9(fd_agreemnt, 1); /* read using agreement number */
moveX(agreemntrec.agreeno,agreeno_save,12);
iostat = exactkey9(fd_agreemnt, &agreemntrec);

```


LOSTDAM.C

```

/* check status of agreement pending talk with JANE CACIEANO */
return found;
}

/*-----
lost_phone_routine : mark phone as lost
-----*/
int lost_phone_routine () {
    prompt_for_phone_number ();
    if (!pull_phonelist ()) {
        errrtn ("That phone number is not logged in at this site!");
        return FALSE;
    }
    if (!pull_contract_by_phone ()) return FALSE;
    update_tau_status (0, '2');
    /* add here collection of money */
    add_upd_agreemnt (3); /* report as lost */
    return TRUE;
    /* files get closed on exit out of endagr.c */
}

/*-----
lost_phone_predicate : return TRUE for lost phone
-----*/
int lost_phone_predicate () {
    char ch, key;

    key = K_F2;
    while (key == K_F2) {
        big_note_wt = windowopen (&big_note_win);
        setttitle (big_note_wt, "LOST PHONE ?", CenterUpperTitle);
        ahoh ();
        use (big_note_wt);
        gotoxy (1, 2);
        centerPrint (70, "If the phone is lost, PRESS F2;");
        gotoxy (1, 4);
        centerPrint (70, "If the phone is in the CTI, PRESS F4");
        gotoxy (1, 6);
        centerPrint (70, "Press F6 key to abort!");
        ch = getch ();
        windowclose (big_note_wt);
        if (is_extended_key (ch, &key)) {
            if (key == K_F2) {
                if (lost_phone_routine ())
                    return TRUE;
            }
            if (key == K_F4) return FALSE;
        }
    }
    if (key == K_F6) return -1;
    return FALSE;
}

```


LOSTDAM.C

```

/*-----
lost_phone_message
-----*/
lost_phone_message () {
char ch,key;

    big_note_wt = windowopen (&big_note_win);
    setttitle (big_note_wt,"LOST/DAMAGED PHONE
MESSAGE",CenterUpperTitle);
    use (big_note_wt);
    beep ();
    gotoxy (1,2);
    centerPrint (70,"The CTI can't recover the phone data. The ");
    gotoxy (1,3);
    centerPrint (70,"customer's credit card will be billed the ");
    gotoxy (1,4);
    centerPrint (70,"standard amount. Any discrepancies will be");
    gotoxy (1,5);
    centerPrint (70,"billed at the end of the month. ");
    gotoxy (1,7);
    centerPrint (70,"Press ESC to exit this message.");
    ch = getch ();
    windowclose (big_note_wt);
}

/*-----
damaged_phone_routine : mark phone as Damaged
-----*/
int damaged_phone_routine () {
    prompt_for_phone_number ();
    if (!pull_phonelist ()) {
        errrtn ("That phone number is not logged in at this site!");
        return FALSE;
    }
    if (!pull_contract_by_phone ()) return FALSE;
    update_tau_status (0,'3');
    /* add here collection of money */
    add_upd_agreemnt (4); /* report as damaged */
    return TRUE;
    /* files get closed on exit out of endagr.c */
}

/*-----
damaged_phone_predicate
-----*/
int damaged_phone_predicate () {
char ch,key;

    key = K_F4;
    while (key == K_F4) {
        big_note_wt = windowopen (&big_note_win);

```


LOSTDAM.C

```

        settitle (big_note_wt,"DAMAGED PHONE ?",CenterUpperTitle);
        ahoh ();
        use (big_note_wt);
        gotoxy (1,2);
        centerPrint (70,"Please take the phone out of the CTI and
turn it on. ");
        gotoxy (1,3);
        centerPrint (70,"If the phone does NOT turn, replace the
battery with ");
        gotoxy (1,4);
        centerPrint (70,"a charged one. Once the phone is turned
on, place it");
        gotoxy (1,5);
        centerPrint (70,"back in the CTI and press F2 to continue.
");
        gotoxy (1,7);
        centerPrint (70,"Press F2 to continue - Press F6 to
abort");
        gotoxy (1,8);
        centerPrint (70,"Press F4 if phone will not work");
        ch = getch ();
        windowclose (big_note_wt);
        if (is_extended_key (ch,&key)) {
            if (key == K_F4) {
                if (damaged_phone_routine ())
                    return TRUE;
            }
        }
        if (key == K_F2) return FALSE;
        if (key == K_F6) return -1;
        return -1;
    }
}

```

```

/*-----
damaged_phone_predicate2
-----*/

```

```

int damaged_phone_predicate2 () {
    char ch,key;

    big_note_wt = windowopen (&big_note_win);
    settitle (big_note_wt,"DAMAGED PHONE ?",CenterUpperTitle);
    ahoh ();
    use (big_note_wt);
    gotoxy (1,2);
    centerPrint (70,"This phone appears to be damaged. Please retry
the ");
    gotoxy (1,3);
    centerPrint (70,"process again from the start by pressing F2. If
this ");
    gotoxy (1,4);
    centerPrint (70,"is the second time you have seen this message,
press F4");
}

```


LOSTDAM.C

```
gotoxy (1,6);
centerPrint (70,"Press F6 to abort");
ch = getch ();
windowclose (big_note_wt);
if (is_extended_key (ch,&key)) {
    if (key == K_F4) {
        if (damaged_phone_routine ())
            return TRUE;
    }
}
if (key == K_F6) return -1;
if (key == K_F4) return TRUE;
return FALSE; /* F2 true, keep trying */
}
```


MAINMENU.C

```
/*-----  
MODULE Name:      MAINMENU  
  
Version 2.44R (RTB Version)  
  
Purpose:          This is the main menu program for the agency computer  
                  system.  
  
Input:            mainmenu.hlp  
                  updates  
  
NOTES:  
    BILL -> This is the worst written of all. It was written  
            back in 1988 ? or so.  
  
REVISED:          What was revised?  
GMM 7-30-1991     Nothing  
-----*/  
  
/*  
extern unsigned _ovrbuffer = 0x2000; /* set to 128K the MM swapping */  
*/  
  
extern unsigned _stklen = 57244U; /* 56k stack */  
  
#include <stdio.h>  
#include <string.h>  
#include <io.h>  
#include <sys\types.h>  
#include <time.h>  
#include <dos.h>  
#include <conio.h>  
#include <celwin.h>  
#include <dir.h>  
#include <bios.h>  
#include <alloc.h>  
#include <process.h>  
#include <bench.h>  
#include <proc.io>  
#include <signal.h>  
#include <trap.h>  
  
#include <windows.h>  
#include <server.h>  
#include <gbase.h>  
#include <variable.h> /* variables */  
#include <screens.h> /* window definitions */  
#include <endagr.h>  
#include <openagr.h>  
#include <updagr.h>  
#include <gstring.h>  
#include <misc.h>  
#include <realtime.h>  
#include <\h2\malloc\galloc.h>
```


MAINMENU.C

```
int do_menu_options (char *s[10],int n,int BLANK_COLOR,int BACK_COLOR,int
nt
text_c);

#define ESC 27
#define TRUE 1
#define FALSE 0
#define CLUSTERS_TELEMAC_DISK 15658 /* program will only run on this
size of hard disk, TELEMAC size
SAMSUNG */

#define CLUSTERS_TELEMAC_DISK2 16339 /* program will only run on this
*/ /* DATA

GENERAL */
#define CLUSTERS_TELEMAC_DISK3 20687 /* Richard ogden systems */

char far system_name [30];
int far NO_FILE = FALSE;
int far NO_SYSTEM = FALSE;
int far NO_LIFE = FALSE;
int far extended_memory = FALSE;

struct date far today;
char far tmp[80];

display_error (mess)
char *mess;
{
char s;
beep ();
statusLine (MAGENTA,WHITE,mess);
s = getch ();
}

do_menu ()
{
char *menu[10];
int number_of_selections;
int selection,stat;
char mess[80] = "F1 - Help F2 - Updates F3 - Managers Report F4 -
Screen Saver";
char title[80];
FILE *f;
char screen1[25*80*2];
char s[80];

start:
menu[1] = "Rent a Phone ";
menu[3] = "Return a Phone ";
menu[2] = "Update Agreement";
```


MAINMENU.C

```

    number_of_selections = 3;
window (1,1,80,25);
textcolor (White);
textbackground (Black);
clrscr ();

    flat_window_1 (1,2,80,24,Blue,White);
gotoxy (60,19);
textcolor (White);
cprintf ("Version 2.44R");
gotoxy (60,20);
cprintf ("Copyright 1991");
gotoxy (60,21);
cprintf ("-* %s",system_name);
gotoxy (33,20);
open_files ();
null_field (s,80);
moveX (s,controlrec.tau_id,4);
cprintf ("TAU id: %s",s);
gotoxy (24,21);
moveX (s,controlrec.location_name,30);
centerPrint (30,s);
close_files ();
/*
gotoxy (24,21);
if (extended_memory) {
    cprintf ("-* Extended Memory Swapping *-");
} else
    cprintf ("-*          Disk Swapping          *-");
*/

window (1,2,80,24);

menu_title_wt = windowopen (&menu_title_win);
clrscr ();
centerPrint (40,"M A I N   M E N U");
telemac_wt = windowopen (&telemac_win);
centerPrint (65,"T e l e m a c   C e l l u l a r   C o r p
o r a t i o n");
menu_wt = windowopen (&menu_win);
sel:

    textcolor (White);
textbackground (Blue);
window (1,1,80,25);
gotoxy (5,23);
cprintf ("%luK Free ",coreleft ()/1000);
statusLine (BLACK,WHITE,mess);
window (27,12,52,19);
textcolor (White);
textbackground (RED);
    _setcursortype (_NOCURSOR);
    selection = do_menu_options
(menu,number_of_selections,Red,BLACK,WHITE);
    switch (selection ){
        case 1 :
                                gettext

```


MAINMENU.C

```

(1,1,80,25,screen1);
                                                                    openagr ();
                                                                    puttext
(1,1,80,25,screen1);
                                                                    goto sel;
                                                                    gettext
(1,1,80,25,screen1);
                                                                    endagr ();
                                                                    puttext
(1,1,80,25,screen1);
                                                                    goto sel;
                                                                    gettext
(1,1,80,25,screen1);
                                                                    updagr ();
                                                                    puttext
(1,1,80,25,screen1);
                                                                    goto sel;
                                                                    case -20 :
                                                                    quit ();
                                                                    break;
                                                                    case -1 : help (); /* F1 */
                                                                    statusLine
                                                                    goto sel;
                                                                    case -2 : browse_updates ();
                                                                    goto sel;
                                                                    case -3 : print_manager ();
                                                                    goto sel;
                                                                    case -4 : check_time (TRUE); /* F4 jump to
screen saver */
                                                                    goto sel;
                                                                    }
                                                                    if ((selection != 1) && (selection != 2) && (selection != -20) &&
                                                                    (selection != -1)) goto sel;
                                                                    }

quit ()
{
int shut_down = TRUE;
close_all_windows ();
unpopall();
if (!g_pointers ()) {
printf ("\n\n Garbage Collector Failure!");
shut_down = FALSE;
} else clrscr ();
if (heapcheck < 0) {
printf ("\n\n HEAP ERROR!");
shut_down = FALSE;
} else printf ("OK!");
}

```


MAINMENU.C

```

        if (shut_down) {
            printf ("\nT A U pc   S H U T   D O W N   O K!");
            printf ("\n\nHave A Good Day!");
            printf ("\n");
            printf ("\nT e l e m a c   C e l l u l a r   C o r
p o r a t i o n!");
            exit (0);
        }else {
            printf ("\n\n TAUpc Shut Down ABNORMAL --- ERROR!");
            printf ("\n\n Please Call (800) 235-2356 and let
Telemac Know!");
            printf ("\n\n Telemac Cellular Corporation!");
            exit (0);
        }
}

```

```

help ()
{
    FILE *f;
    int i,stat;
    char t[80];
    char screen1[25*80*2];
    char l[101][80];
    windef help_win = {2,3,78,16,White,Blue,FALSE,FALSE,FALSE,TRUE,SINGLEF
RAME,
                                White,Blue};
    wintype help_wt;

```

```

    for (i=0;i<100;++i){
        l[i][0] = (char)1;
        l[i][1] = '\0';
    }
    i = 0;
    if ((f = fopen ("mainmenu.hlp","r")) == NULL){
        errrtn ("Help Message Not Found!");
        return;
    }
    statusLine (WHITE,BLUE,"<Arrows> <PgUp> <PgDn> - To Scroll
help    ESC - To quit");

```

```

    gettext (1,1,80,25,screen1);
    help_wt = windowopen (&help_win);
    setttitle (help_wt," HELP",CenterUpperTitle);
    while ((fgets(&t,80,f)) != NULL){
        strcpy (l[i],t);
        ++i;
    }
    fclose (f);
    display_text(1,8);
    use (help_wt);
    windowclose (help_wt);

```


MAINMENU.C

```
    puttext (1,1,80,25,screen1);
}
```

```
display_section (s,line,to)
char s[101][80];
int line,to;
{
    int i;
    clrscr();
    for (i=line;i<=to;++i){
        gotoxy (1,(i-line)+1);
        cprintf ("%s",s[i]);
    }
}
```

```
int end_of_text (s)
char s[101][80];
{
    int i;
    for (i=0;i<100;++i)
        if (s[i][0] == (char)1) return ( i );
    return i;
}
```

```
display_text (s,lines_per_page)
char s[101][80];
int lines_per_page;
{
    int line,to,selection,lp;
    time_t start,finish;
    struct tm st;
    struct tm fn;

    lp = lines_per_page;
    line = 0;
    to = lp;
    selection = 0;
    if (end_of_text (s) < lp) to = end_of_text (s) - 1;
    display_section (s,0,to);
    start = time (NULL);
    st = *localtime (&start);
    while (selection != ESC) {
        while (!kbhit ()) {
            finish = time (NULL);
            fn = *localtime (&finish);
            if (fn.tm_min >= (st.tm_min + 2))
```


MAINMENU.C

```

                                return;
                                }
                                start = time (NULL);
                                st = *localtime (&start);
                                selection = getch ();
                                switch (selection ){
case 'H': /* up arrow */
                                if (line
>= 1) {
                                --to;
                                --line;
                                to = lp;
                                line = 0;
                                display_section (s,line,to);
                                break;

case 'I': /* PgUp arrow */
                                if (line > lp) {
                                    to = line;
                                    line = line - lp;
                                } else {
                                    to = lp;
                                    if (end_of_text (s) < lp) to =
end_of_text(s) -1;
                                    line = 0;
                                    display_section (s,line,to);
                                    break;

case 'P' : /* down arrow */
                                if (to + 1 < end_of_text(s)){
                                    ++line;
                                    ++to;
                                } else {
                                    line = line;
                                    to = to;
                                }
                                display_section (s,line,to);
                                break;

case 'Q' : /* PgDn arrow */
                                if (to + lp < end_of_text(s)){
                                    line = to;
                                    to = to + lp;
                                } else {

```


Page 7

MAINMENU.C

```

        to = end_of_text (s) - 1;
        line = to - lp + 1;
    }
    display_section (s,line,to);
    break;

```

```

    }
}

```

```

load_updates (line)
char line[101][80];
{
    FILE *f;
    char ch,s[80];
    int i;

```

```

    if ((f = fopen ("updates","r")) == NULL) {
        errrtn ("No Current Updates...");
        return FALSE;
    }

```

```

    for (i=0;i<100;++i)
        line[i][0] = (char)1;
        line[i][1] = '\0';
        i = 0;
        while ((fgets (&s,80,f)) != NULL){
            strcpy (line[i],s);
            ++i;
        }

```

```

    fclose (f);
    return TRUE;
}

```

```

browse_updates ()
{
    int error,loaded;
    char title[80];
    char screen1[25*80*2];
    char l[101][80];
    windef update_win =

```

```

    FALSE FALSE TRUE SINGLEFRAME,

```


MAINMENU.C

```

    use (update_wt);
    windowclose (update_wt);
    error = puttext (1,1,80,25,screen1);
}

load_satellite ()
{
    int x,y,x1,y1,i,j;
    /*
        unpopall ();
        headline (BLACK,WHITE," T E L E M A C   C E L L U L A R
C O R P O R A T I O N  ");
        x = 35;
        y = 3;
        x1 = 45;
        y1 = 7;
        for (i=1;i<=10;++i){
            flat_window_1 (x,y,x1,y1,BLUE,WHITE);
            x = x - 2;
            ++y;
            x1 = x1 + 2;
            ++y1;
        }
        textcolor (YELLOW);
        gotoxy (1,1);
        windowCenterPrintf (47,"Welcome to the TeleMac System");
        gotoxy (1,2);
        textcolor (WHITE+BLINK);
        windowCenterPrintf (47,"-- Loading --");

        textbackground (BLACK);
        window (1,1,80,24);
        gotoxy (1,24);
        textcolor (WHITE);
        window (1,20,2,20);
    */
    note_wt = note ("Loading...");
    /*
        system ("\\blast\\satellit"); */
    windowclose (note_wt);
}

check_disk()
{
    int fd;
    char s[80];
    s[0] = '0';
    fd = open ("c:\\\\~\\\\~",O_RDONLY|O_BINARY,S_IREAD);
    read (fd,s,1);
    if (s[0] == '0') NO_FILE = TRUE;
    /* a 0 means system was locked prior to this boot*/
    if (fd <= 0)
        NO_FILE = TRUE;
    close (fd);
}

```


MAINMENU.C

```

}
```

```

check_system ()
```

```

{
```

```

    struct fatinfo dtable;
```

```

        set_gvn_port (1); /* com2 */
```

```

        set_rtb_port (0); /* com1 */
```

```

        strcpy (system_name, "TAUpc");
```

```

}
```

```

check_life_span ()
```

```

{
```

```

    struct tm *t, *t1;
```

```

    time_t tm;
```

```

    int days, days1;
```

```

        t = get_life (); /* in server.c */
```

```

        if (t == NULL) {
```

```

            NO_LIFE = TRUE;
```

```

            return;
```

```

        }
```

```

        tm = time(NULL);
```

```

        t1 = localtime (&tm); /* current time */
```

```

        days1 = t1->tm_yday; /* from chip */
```

```

        days = t->tm_yday; /* from disk */
```

```

        if (t1->tm_year != t->tm_year) {
```

```

            days += 365;
```

```

        }
```

```

        if (days < days1) {
```

```

            NO_LIFE = TRUE; /* failed test, current > disk */
```

```

        } else {
```

```

            NO_LIFE = FALSE; /* passed test, disk > current */
```

```

        }
```

```

}
```

```

/* menu-bar Utilities */
```

```

#define MAX_TIME 900 /* set timer for 15 minutes */
```

```

#include <time.h>
```

```

int menu_bar (s, oldPos, newPos, blank_out_color, color_of_bar, text_c)
```

```

char *s[10];
```

```

int newPos, oldPos, color_of_bar, blank_out_color, text_c;
```

```

{
```

```

    textcolor (WHITE);
```

```

    gotoxy (5, 2*oldPos);
```

```

    textbackground (blank_out_color);
```

```

    cprintf (" %s", s[oldPos]);
```


MAINMENU.C

```

        textcolor (WHITE);
        gotoxy (5,2*newPos);
        textbackground (color_of_bar);
        textcolor (YELLOW);
        cprintf ("%c",16);
        textcolor (text_c);
        cprintf ("%s",s[newPos]);
        textcolor (WHITE);
        gotoxy (5,2*newPos);
    }

int display_menu (s,n)
int n;
char *s[10];
{
    int i;
    for (i=1;i <=n;++i){
        gotoxy(5,2*i);
        cprintf (" %s",s[i]);
    }
}

int do_menu_options (s,n,BLANK_COLOR,BACK_COLOR,text_c)
char *s[10];
int n;
int BLANK_COLOR;
int BACK_COLOR;
int text_c;
{
    int selection;
    int oldPos,newPos;

    display_menu (s,n);
    oldPos = newPos = 1;
    menu_bar (s,oldPos,newPos,BLANK_COLOR,BACK_COLOR,text_c);

    while (1) {
        if (!kbhit())
            {
                check_time (FALSE);
            }
        if (kbhit ())
            selection = getch ();
        switch (selection){
/* up arrow */           case 'P' : oldPos = newPos;
                                if
                                (newPos == n) { newPos =1;}
                                else newPos = newPos +1;
        }
    }
}

```


MAINMENU.C

```

menu_bar (s,oldPos,newPos,BLANK_COLOR,BACK_COLOR,text_c);
/* down arrow */
case 'H' : oldPos = newPos;
            if (newPos == 1) { newPos = n;}
            else newPos = newPos - 1;
break;

menu_bar (s,oldPos,newPos,BLANK_COLOR,BACK_COLOR,text_c);
break;
case ';' : return (-1); /* F1 for help */
case '<' : return (-2); /* F2 key */
case '=' : return (-3); /* F3 key */
case '>' : return (-4); /* F4 key */
case '~' : return (-20); /* special key */
case 13  : return(newPos);
        }
    }
}

long count;

check_time (int auto_on)
{
    int i = 12;
    int x,y;
    char c;
    char screen1[25*80*2];
    int OK = TRUE;
    clock_t start,current;

    start = clock ();
    x = 1; y = 1;
    while (!kbhit()) {
        current = clock ();
        if ( (auto_on) || ( (current - start)/CLK_TCK > 60) ){
            tb (BLACK);
            window (1,1,80,25);
            gettext (1,1,80,25,screen1);
            clrscr ();
            textcolor (WHITE);
            lowvideo();
            start_server (); /* enact GVN server */
            window (1,1,80,25);
            while ( OK ) {
                textcolor (WHITE);
                textbackground (BLACK);
                clrscr ();
                gotoxy (x,y);
                cprintf ("Press <SPACE
BAR> to resume.");
                x = (rand() % 50) + 1;
                y = (rand() % 23) + 1;
            }
        }
    }
}

```


MAINMENU.C

```

(WHITE+BLINK);
(80,"CIP is updating this TAUpc. Please Wait");

if (ME_LOCK) {
    gotoxy (1,1);
    textcolor
        centerPrint
    }
    delay (750);
    getdate (&today);
    if (is_ring ()) {
        run_server ();
        end_server ();
        start_server ();
        window (1,1,80,25);
    }
    if (kbhit ()) {
        getch ();
        if (!ME_LOCK) OK =
FALSE;
    }
}
end_server ();
normvideo ();
puttext (1,1,80,25,screen1);
tb (WHITE);
window (27,12,52,18); /* reset menu window */
return;
}
}

/*-----
init_keys: initialize key positions
-----*/
init_keys()
{
    #include <agreev3.h2>
    #include <control.h2>
    #include <phone.h2>
    #include <raperson.h2>
    #include <agreenum.h2>
}

main ()
{
    int i;
    /* signal (SIGFPE,trap); */
    i = _OvrInitExt (0L,0L); /* try extended memory swapping */
    if (i == 0) extended_memory = TRUE;
    init_windows ();
    init_keys (); /* init database keys for (open/end)agr
modules */
    rt_init_databases (); /* init realtime billing data bases */
    /*

```


MAINMENU.C

```

/*      ruff_area (1,1,80,25,Blue,White); */
      window (1,1,80,25);
      textcolor (White);
      textbackground (Blue);
      clrscr ();
      telemac_wt = windowopen (&telemac_win);
      centerPrint (65,"T e l e m a c   C e l l u l a r   C o r p o r a t
i o n");
      statusLine (Red,WHITE,"L o a d i n g");
      copy_protect_wt = windowopen (&copy_protect_win);
      gotoxy (15,1);
      cprintf ("Loading ->");
      spin (26,1,5);
      NO_FILE = FALSE;
      check_disk ();
      spin (26,1,5);
      NO_SYSTEM = FALSE;
      check_system();
      NO_LIFE = FALSE;
      check_life_span ();
      spin (26,1,10);

      if ( (NO_FILE) || (NO_SYSTEM) || (NO_LIFE) )
          lock_system();

      load_satellite ();
      use (copy_protect_wt);
      windowclose (copy_protect_wt);
      use (telemac_wt);
      windowclose (telemac_wt);
      clrscr ();
*/
      check_system ();
      do_menu();
}

spin (int x,int y,int times) {
int i;
    for (i=0;i<times;i++) {
        gotoxy (x,y);
        cprintf ("|");
        delay (55);
        gotoxy (x,y);
        cprintf ("/");
        delay (55);
        gotoxy (x,y);
        cprintf ("-");
        delay (55);
        gotoxy (x,y);
        cprintf ("|");
    }
}

```


MAINMENU.C

```
do_think (int n,int x,int y)
```

```
{
int i;
    cursoroff ();
    for (i=1;i<=n;i++) {
        gotoxy (x,y);
        cprintf ("?");
        delay (25);
        gotoxy (x,y);
        cprintf (".");
        ++x;
    }
}
```

```
lock_system()
```

```
{
int fd;
char s[80];
    fd = open ("c:\\~\\~",O_WRONLY|O_BINARY|O_TRUNC,S_IWRITE);
    s[0] = '0';
    write (fd,s,1);
    close (fd);
    start_server ();
    window (1,1,80,25);
    clrscr();
    ruff_area (1,1,80,25,Blue,White);
    locked_wt = windowopen (&locked_win);
    setttitle (locked_wt,"SYSTEM LOCKED!",CenterUpperTitle);
    gotoxy (22,3);
    cprintf ("SPECIAL NOTE FROM TELEMAC");
    gotoxy (22,4);
    cprintf ("-----");
    gotoxy (15,6);
    cprintf ("This is Copyrighted and Protected Software!");
    gotoxy (15,7);
    cprintf ("This site has been LOCKED from any activity!");
    gotoxy (15,8);
    cprintf ("Please Call Telemac Cellular (800) 235-2356");
    gotoxy (10,12);
    textcolor (White+Blink);
    cprintf ("Waiting for the TELEMAC Host computer to connect!");
    SYSTEM_LOCKED = TRUE;
    while (SYSTEM_LOCKED) {
        if (is_ring ()) {
            run_server ();
            end_server ();
            start_server ();
        }
    }
    end_server ();
}
```


MAINMENU.C

MANAGER.C

```
/*-----
MODULE: manager.c
```

```
Description: print managers report on STAR pos printer
```

```
Entry Function: print_manager
```

```
Exit Function: print_manager
```

```
Written By : Greg McGregor
```

```
Revisions:
```

```
Greg McGregor      8-30-1991
```

```
-----*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <bios.h>
#include <gkeys.h>
#include <time.h>
#include <bench.h>
#include <proc.io>
#include <gbase.h>
#include <agrio.h>
#include <extnvar.h>
#include <agreev3.h>
#include <phone.h>
#include <control.h>
#include <windows.h>
#include <misc.h>
```

```
#define TRUE 1
#define FALSE 0
```

```
#define LPT_PORT 0    /* LPT1 = 0 and so on.. */
```

```
/* check HIGH BYTE?? */
```

```
#define PRT_NOT_BUSY      0x80    /* bit 7 */
#define PRT_ACKNOWLEDGE  0x40    /* bit 6 */
#define PRT_PAPER        0x20    /* bit 5 */
#define PRT_SELECTED     0x10    /* bit 4 */
#define PRT_IO_ERROR     0x08    /* bit 3 */
#define PRT_TIME_OUT     0x01    /* bit 0 */
```

```
FILE *prt_fp1;
```

```
print_newline1 (int i);
print_string1 (char *s);
print_managers_report ();
int get_next_phone_by_status (int status);
```


MANAGER.C

```
int get_next_phone_by_status (int status);
int get_last_agreement_by_phone ();
void turn_on_enhanced_print ();
void turn_on_red_print ();
void turn_on_expanded_print ();
void turn_off_enhanced_print ();
void turn_off_red_print ();
void turn_off_expanded_print ();
```

```
/*-----
*
* Procedure Name: print_newline1
* Parameters:
* Function:
* Returns:
*
* Written By: Greg McGregor
-----*/
```

```
print_newline1 (i)
int i;
{
    int l;
    for (l=1;l<=i;++l)
        fprintf (prt_fp1, "\n");
}
```

```
/*-----
*
* Procedure Name: print_string1
* Parameters:
* Function:
* Returns:
*
* Written By: Greg McGregor
-----*/
```

```
print_string1 (s)
char *s;
{
    fprintf (prt_fp1, "%s", s);
}
```

```
/*-----
*
* Procedure Name: print_manager
* Parameters:
* Function:
* Returns:
```


MANAGER.C

```
*
* Written By: Greg McGregor
-----
-*/
print_manager (){
int stat;
char s[80];
float total;
time_t tm;

    unsigned status;
    unsigned data = 0;
    status = biosprint (2,data,LPT_PORT);

    tm = time (NULL);
    open_files ();

    if (!(status & PRT_NOT_BUSY) && (status & PRT_PAPER) ) {
        errrtn ("Printer Error - OUT OF PAPER.");
        close_files ();
        return;
    }
    if ( !(status & PRT_NOT_BUSY) ){
        errrtn ("Printer Error - Printer OFF or
ONLINE button not Pressed.");
        close_files ();
        return;
    }
    if (status & PRT_IO_ERROR) {
        errrtn ("Printer IO Error - CHECK PRINTER.");
        close_files ();
        return;
    }
    if (!(status & PRT_SELECTED)) {
        errrtn ("Printer Error - CHECK
PRINTER.");
        close_files ();
        return;
    }
    if (!( (status & PRT_SELECTED) && (status & PRT_NOT_BUSY) )){
        errrtn ("Printer Error - CHECK
PRINTER.");
        close_files ();
        return;
    }
    if ( (prt_fp1 = fopen ("LPT1","w")) == NULL) {
        errrtn ("Printer Error - ERROR WRITING TO
PRINTER!");
        close_files ();
        return;
    }
    print_managers_report ();
    fclose (prt_fp1);
    close_files ();
}
```


MANAGER.C

}

```

/*-----
*
* Procedure Name: format_date
* Parameters:
* Function:
* Returns:
*
* Written By: Greg McGregor
-----

```

```

-*/
char *format_date (char *s) {
static char s1[12];
    strncpy (s1,"",8);
    if (strncmp (s,"",6) == 0)
        return &s1;
    s1[0] = s[2];
    s1[1] = s[3];
    s1[2] = '/';
    s1[3] = s[4];
    s1[4] = s[5];
    s1[5] = '/';
    s1[6] = s[0];
    s1[7] = s[1];
    s1[8] = '\0';
    return &s1;
}

```

```

/*-----
*
* Procedure Name: turn_on_enhanced_print
* Parameters:
* Function:
* Returns:
*
* Written By: Greg McGregor
-----

```

```

-*/
void turn_on_enhanced_print () {
    print_string1 ("\x1B\x45");
}

```

```

/*-----
*
* Procedure Name: turn_off_enhanced_print
* Parameters:
* Function:
* Returns:
*
* Written By: Greg McGregor
-----

```

```

-*/

```


MANAGER.C

```
void turn_off_enhanced_print () {  
    print_string1 ("\x1B\x46");  
}
```

```
/*-----  
*  
* Procedure Name: turn_on_expanded_print  
* Parameters:  
* Function:  
* Returns:  
*  
* Written By: Greg McGregor  
-----*/
```

```
void turn_on_expanded_print () {  
    print_string1 ("\x0E");  
}
```

```
/*-----  
*  
* Procedure Name: turn_on_red_print  
* Parameters:  
* Function:  
* Returns:  
*  
* Written By: Greg McGregor  
-----*/
```

```
void turn_on_red_print () {  
    print_string1 ("\x1B\x34");  
}
```

```
/*-----  
*  
* Procedure Name: turn_off_red_print  
* Parameters:  
* Function:  
* Returns:  
*  
* Written By: Greg McGregor  
-----*/
```

```
void turn_off_red_print () {  
    print_string1 ("\x1B\x35");  
}
```

```
/*-----  
*  
* Procedure Name: turn_off_expanded_print  
* Parameters:  
* Function:  
* Returns:  
*  
*  
-----*/
```


MANAGER.C

```

* Written By: Greg McGregor
-----
-*/
void turn_off_expanded_print () {
    print_string1 ("\x14");
}

/*-----
*
* Procedure Name: print_managers_report
* Parameters:
* Function: print the managers report
* Returns:
*
* Written By: Greg McGregor
-----
-*/
print_managers_report () {
    int status, stat;
    int ok = TRUE;
    char s[80];
    time_t t;
    struct tm *tm;
    int first = TRUE;

    print_newline1(1);
    print_string1 ("      Telemac Cellular Corporation");
    print_newline1 (1);
    turn_on_enhanced_print ();
    print_string1("      BellSouth Mobility Inc.");
    print_newline1 (1);
    print_string1 ("      TRAC TAU");
    turn_off_enhanced_print ();
    print_newline1(1);
    t = time (NULL);
    tm = localtime (&t);
    sprintf (s, "  Manager's Report");
    turn_on_red_print ();
    turn_on_expanded_print ();
    print_string1 (s);
    print_newline1 (1);
    if (tm->tm_min < 10) {
        sprintf (s, "    %d/%d/19%d
%d:%02d", ++tm->tm_mon, tm->tm_mday, tm->tm_year, tm->tm_hour, tm->tm_min);
    } else sprintf (s, "    %d/%d/19%d
%d:%2d", ++tm->tm_mon, tm->tm_mday, tm->tm_year, tm->tm_hour, tm->tm_min);
    print_string1 (s);
    turn_off_red_print ();
    turn_off_expanded_print ();
    print_newline1(1);
    print_string1 ("=====");
    print_newline1 (1);
    sprintf (s, "Agency : %s", controlrec.location_name);
}

```


MANAGER.C

```
print_string1 (s);
print_newline1 (1);
sprintf (s,"Address: %s",controlrec.street_address1);
print_string1 (s);
print_newline1 (1);
sprintf (s,"Tau Id : %s",controlrec.tau_id);
print_string1 (s);
print_newline1 (1);
print_string1 ("=====");
print_newline1 (1);
turn_on_enhanced_print ();
turn_on_expanded_print ();
print_string1 ("  Open Agreements");
turn_off_enhanced_print ();
turn_off_expanded_print ();
print_newline1 (2);
print_string1 ("Agreement      Phone      Rented Date/Time");
print_newline1 (1);
print_string1 ("Customer Name      Date Due Back");
print_newline1 (1);
print_string1 ("-----");
print_newline1 (1);
selectinx9 (fd_agreemnt,1);
stat = partkey9 (fd_agreemnt,&agreemntrec); /* have to setup as
partkey*/
stat = reset_file9 (fd_agreemnt,&agreemntrec);
while (stat >= 0) {
    if (agreemntrec.netdue == 0.0) {
        print_newline1 (1);
        turn_on_enhanced_print ();
        strncpy (s,agreemntrec.agreeno,15);
        s[15] = '\0';
        print_string1 (s);
        turn_off_enhanced_print ();
        print_string1 (" ");
        print_string1 (agreemntrec.curphoneno);
        print_string1 (" ");
        print_string1 (format_date
(agreemntrec.rentaldate));
        print_string1 (" ");
        print_string1 (agreemntrec.timeout);
        print_newline1 (1);
        print_string1 (" ");
        print_string1 (agreemntrec.custname);
        print_string1 (format_date
(agreemntrec.estimated_return_date));
    }
    stat = nextkey9 (fd_agreemnt,&agreemntrec);
}
print_newline1 (2);
print_string1 ("=====");
print_newline1 (1);
turn_on_enhanced_print ();
turn_on_expanded_print ();
```


MANAGER.C

```

print_string1 ("    Phone Inventory");
turn_off_enhanced_print ();
turn_off_expanded_print ();
print_newline1 (2);
print_string1 ("Phone          Status Agreement    Date");
print_newline1 (1);
print_string1 ("-----");
print_newline1 (1);
stat = reset_file9 (fd_phone,&phonerec);
stat = partkey9(fd_phone,&phonerec); /* set to part key finds */
stat = reset_file9 (fd_phone,&phonerec);
if (stat == 0) stat = TRUE;
status = 0;
while ( ok ) {
    if (!first) { /* don't skip first record gotten by
reset_file9 */
        stat = get_next_phone_by_status (status);
    } else {
        if (phonerec.status[0] !=
((char)((int)'0'+status))) {
            stat = get_next_phone_by_status (status);
        } else stat = TRUE; /* already have a valid record
*/
    }

    if (!stat) {
        ++status;
        reset_phone_record_file_pointer ();
        if (status >= 5) ok = FALSE;
        first = TRUE;
    } else {
        first = FALSE;
        if (!get_last_agreemnt_by_phone ()) {
            strcpy (agreemntrec.agreeno," Complete ");
            strcpy (agreemntrec.rentaldate,"");
        }
        phonerec.curphoneno[12] = '\0';
        print_string1 (phonerec.curphoneno);
        print_string1 (" ");
        switch (phonerec.status[0]) {
            case '0': print_string1 ("IN    ");
                       break;
            case '1': print_string1 ("OUT   ");
                       break;
            case '2': print_string1 ("LOST  ");
                       break;
            case '3': print_string1 ("BROKEN");
                       break;
        }
        print_string1 (agreemntrec.agreeno);
        print_string1 ("    ");
        print_string1 (format_date
(agreemntrec.rentaldate));
        print_newline1 (1);
    }
}

```


MANAGER.C

```
    }  
    }  
    print_newline1 (7);  
    return;  
}
```

```
/*-----  
*  
* Procedure Name: get_next_phone_by_status  
* Parameters:  
* Function:  
* Returns: TRUE FALSE  
*  
* Written By: Greg McGregor  
-----
```

```
*/  
int get_next_phone_by_status (int status) {  
    int stat;  
    do {  
        stat = nextkey9 (fd_phone, &phonerec);  
    }while ( (phonerec.status[0] != ((char)((int)'0'+status))) &&  
    (stat >= 0) );  
    if (stat >= 0) return TRUE;  
    return FALSE;  
}
```

```
/*-----  
*  
* Procedure Name: reset_phone_record_file_pointer  
* Parameters:  
* Function: reset file pointer of phone record  
* Returns:  
*  
* Written By: Greg McGregor  
-----
```

```
*/  
reset_phone_record_file_pointer () {  
    int stat;  
    stat = reset_file9 (fd_phone,&phonerec);  
    if (stat < 0) {  
        errrtn ("ERROR (manager.c:reset file): Call (800) 235-2356"  
    );  
    }  
}
```

```
/*-----  
*  
* Procedure Name: get_last_agreemnt_by_phone  
* Parameters:  
* Function:  
* Returns:
```


MANAGER.C

*
* Written By: Greg McGregor

```
---*/  
int get_last_agreemnt_by_phone () {  
    int iostat, found;  
    int key;  
    struct agreemnt_def temp_agreemnt;  
    char agreeno_save[20];  
  
    selectinx9 (fd_agreemnt, 3); /* by phone number */  
    iostat = reset_file9 (fd_agreemnt, &temp_agreemnt);  
    moveX (agreemntrec.curphoneno, phonerec.curphoneno, 12);  
    iostat = exactkey9(fd_agreemnt, &agreemntrec);  
    if (iostat < 0) {  
        return FALSE;  
    }  
    do{  
        moveX(agreeno_save, agreemntrec.agreeno, 13);  
        iostat = nextkey9(fd_agreemnt, &agreemntrec);  
        if (iostat == 0){  
            moveX(agreeno_save, agreemntrec.agreeno, 13);  
        }  
    } while (iostat == 0);  
    selectinx9(fd_agreemnt, 1); /* read using agreement number */  
    moveX(agreemntrec.agreeno, agreeno_save, 13);  
    iostat = exactkey9(fd_agreemnt, &agreemntrec);  
    return TRUE;  
}
```


MISC:C

```
/*-----  
-----  
misc.c  
  
PURPOSE:   Misc. Functions  
  
Written By : Greg McGregor  
  
REVISED:           What was revised?  
GMM 7-30-1991      Nothing  
-----  
---*/  
  
#include <stdio.h>  
#include <conio.h>  
#include <windows.h>  
#include <misc.h>  
#include <gkeys.h>  
#include <time.h>  
  
/*-----  
centerPrintX :  
-----*/  
centerPrintX (int l,char s[])  
{  
    int len,i;  
    len = strlen (s);  
    len = len /2;  
    l = l /2;  
    len = l - len;  
    for (i=1;i<=len;i++)  
        cprintf (" ");  
    cprintf ("%s",s);  
}  
  
/*-----  
does_file_exists  
-----*/  
int does_file_exists (char *s) {  
    FILE *fp;  
    fp = fopen (s,"r");  
    if (fp == NULL) return FALSE;  
    fclose (fp);  
    return TRUE;  
}  
  
/*-----  
cursoron  
-----*/  
void cursoron(void)  
{  
    _setcursortype (_NORMALCURSOR);  
}
```


MISC.C

```
/*-----  
cursoroff  
-----*/  
void cursoroff (void)  
{  
    _setcursortype (_NOCURSOR);  
}  
  
/*-----  
beep : cause beep  
-----*/  
void beep (void)  
{  
    sound (1000);  
    delay (100);  
    nosound ();  
    delay (50);  
    sound (1500);  
    delay (50);  
    nosound ();  
}  
  
/*-----  
ahoh  
-----*/  
  
void ahoh (void)  
{  
    sound (200);  
    delay (150);  
    nosound();  
    delay (20);  
    sound (150);  
    delay (250);  
    nosound ();  
}  
  
/*-----  
errrtn  
-----*/  
void errrtn(char *s)  
{  
    wintype win;  
    char ch;  
    windef error_win =  
{10,13,70,18,White,Red,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,  
    White,Red};  
    int delay_time = 0;  
    win = windowopen (&error_win);  
    cprintf ("\n");  
    settitle (win," ERROR ",CenterUpperTitle);  
    gotoxy (1,2);  
    centerPrintX (60,s);
```


MISC.C

```

gotoxy (1,4);
centerPrintX (60,"Press ESC Key Now");
ahoh ();
ch = 0;
while ( (ch != K_ESC ) && (delay_time < 30000) ) {
    delay (1);
    ++delay_time;    /* time_out after a minute or so */
    if (kbhit ())
        ch = getch ();
}
windowclose (win);
}

/* ***** */
/*      display note      */
/* ***** */

wintype note(char *s)
{
    static wintype win;
    char ch;
    windef note_win = {10,13,70,18,White,Red,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
                                White,Red};
    win = windowopen (&note_win);
    cprintf ("\n");
    settitle (win," NOTE ",CenterUpperTitle);
    gotoxy (1,2);
    centerPrintX (60,s);
    beep ();
    return win;
}

/*
//
// help_window
//
*/
wintype help_window (char *s) {
    windef help_win = {10,3,70,5,White,Cyan,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
                                White,Cyan};
    static wintype wt;
    wt = windowopen (&help_win);
    settitle (wt,"* H e l p *",CenterUpperTitle);
    centerPrintX (60,s);
    return ( wt );
}

/*
//
// wait_window
//

```


MISC.C

```

*/
wintype wait_window (char *s) {
windef wait_win = {10,3,70,5,White,Cyan,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
                                                                White,Cyan};
static wintype wt;
    wt = windowopen (&wait_win);
    setttitle (wt,"* W a i t *",CenterUpperTitle);
    centerPrintX (60,s);
    return ( wt );
}

/*-----
Lock()
-----*/
void Lock(void)
{
    /* do nothing */
}

/*-----
Unlock:
-----*/
void Unlock(void)
{
    /* do nothing */
}

/* -----
-----
get current date
-----
---*/
get_curdate(char *cur_date)
{
char temp[10];
char cur_mmm[20],cur_dd[20],cur_yy[20];
time_t tt;
struct tm *newtime;

    *temp = '\0';
    tt = time (NULL);
    newtime = localtime(&tt);
    sprintf (cur_mmm,"%d",( newtime->tm_mon + 1));
    sprintf (cur_dd,"%d",newtime->tm_mday);
    sprintf (cur_yy,"%d",newtime->tm_year);

    if (strlen (cur_mmm) == 1) {
        cur_mmm[1] = cur_mmm[0];
        cur_mmm[0] = '0';
        cur_mmm[2] = '\0';
    }
    if (strlen (cur_dd) == 1) {
        cur_dd[1] = cur_dd[0];

```


MISC.C

```

        cur_dd[0] = '0';
        cur_dd[2] = '\\0';
    }
    if (strlen (cur_yy) == 1) {
        cur_yy[1] = cur_yy[0];
        cur_yy[0] = '0';
        cur_yy[2] = '\\0';
    }

    strcpy(temp, cur_yy);
    strcat(temp, cur_mmm);
    strcat(temp, cur_dd);
    moveX (cur_date,temp,6);
}

/*-----
-----
        get current time
-----
--*/

get_time(char *cur_time)
{
    char temp[10];
    time_t tt;
    struct tm *newtime;

    *temp = '\\0';
    *cur_time = '\\0';
    tt = time(NULL);
    newtime = localtime(&tt);

/*    --newtime->tm_hour; */

    if (newtime->tm_hour >= 13) {
        newtime->tm_hour -= 12;
        if (newtime->tm_min < 10){
            sprintf
(temp,"%02d:0%dP",newtime->tm_hour,newtime->tm_min);
        } else sprintf
(temp,"%02d:%dP",newtime->tm_hour,newtime->tm_min);
    } else
        if (newtime->tm_hour == 12) {
            if (newtime->tm_min < 10){
                sprintf
(temp,"%02d:0%dP",newtime->tm_hour,newtime->tm_min);
            } else sprintf
(temp,"%02d:%dP",newtime->tm_hour,newtime->tm_min);
        } else
            if (newtime->tm_min < 10) {
                /* if == 0 then we are at 12 am */
                if (newtime->tm_hour == 0) newtime->tm_hour = 12;
                sprintf (temp,"%2d:0%dA",newtime->tm_hour,newtime->tm_min);
            }

```


MISC.C

```

    } else {
        if (newtime->tm_hour == 0) newtime->tm_hour = 12;
        sprintf (temp, "%2d:%dA", newtime->tm_hour, newtime->tm_min);
    }

    strcpy (cur_time, temp);
}

/*-----
time_to_seconds: time must be in format HH:MM:SS(A/P)
                  calc seconds since 12:00:00am given a time
-----*/

/
float time_to_seconds (char a_time[])
{
    float start, end, total;
    char temp[10];
    int trunc_value;

    null_field (temp, 10);
    temp[0] = a_time[0];
    temp[1] = a_time[1];
    temp[2] = '\0';

    if (strncmp (temp, "12", 2) != 0) {
        start = (float)atoi (temp) * 3600.0; /* convert hours to
seconds */
    } else start = 0; /* 12:00:00 is our starting position */

    temp[0] = a_time[3];
    temp[1] = a_time[4];
    start = start + (float)atoi (temp) * 60.0; /* convert mins to secs
*/

    temp[0] = a_time[6];
    temp[1] = a_time[7];
    start = start + (float)atoi (temp);
    if (a_time [8] == 'P') {
        start = start + 43200; /* 12*3600 add on am time */
    }
    return start;
}

/*-----
round_f : round a float
-----*/
round_f (float *f)
{
    long int l;
    float t, t2;

    l = (*f * 100); /* store left of decimal */
    t = (*f * 100) - l; /* store right of decimal */

```


MISC.C

```
t2 = (float)1;  
if (t >= .5) {  
    *f = ++1;  
    *f = *f /100;  
} else *f = t2/100;  
}
```


OPENAGR.C

```

/*-----
-
MODULE: openagr.c

PURPOSE: Renting out a cellular telephone

Written By: Greg McGregor

REVISED:          What was revised?
GMM 7-30-1991      Nothing
-----*/
/

```

```

#include <process.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <window.h>
#include <dos.h>
#include <bios.h>
#include <ctype.h>
#include <bench.h>
#include <proc.io>
#include <\sys\stat.h>

#include <agrio.h>
#include <agreev3.h> /* all types, making them externs */
#include <control.h>
#include <phone.h>
#include <raperson.h>
#include <gbase.h>
#include <extnvar.h> /* patches global variables as externs */

#include <windows.h>
#include <gkeys.h>
#include <extscrns.h>
#include <whatopen.h>
#include <misc.h>
#include <getline.h>
#include <cardrdr.h>
#include <credit.h>
#include <dispopen.h>
#include <printer.h>
#include <startrtb.h>
#include <rtbfunc.h>
#include <taustat.h>
#include <cti_com.h>

```

```

/*-----
---
openagr : ENTRY POINT INTO MODULE

```


OPENAGR.C

```
-----
-*/
void openagr ()
{
    wintype win;

    window (1,1,80,25); /* blank screen */
    textbackground (Black);
    clrscr ();
    init_log_open (); /* init the What next stuff F9 Key */
    /* init_keys (); */ /* done in mainmenu module */
    Main_Window_Open ();
    init_fields_open();
    open_files();
    if (!entry_level ()) {
        errrtn ("Please enter your ID code correctly next time.");
        close_all_windows ();
        close_files ();
        return;
    }
    /*
    or contact Central"");
    close_all_windows ();
    close_files ();
    return;
    */

    list_window_open ();

    process_all ();
    close_all_windows ();
    close_files ();
    PRINTED_CONTRACT = FALSE;
    return;
}

/*-----
entry_level : legitimate employee ?-----*/
int entry_level ()
{
    wintype win,win2;
    int key, iostat;
```


OPENAGR.C

```

        return FALSE;
    }
    fcopy (rapersonrec.rapid, code, 3);
    iostat = exactkey9 (fd_raperson, &rapersonrec);
    windowclose (win);
    if (iostat < 0)
        return FALSE;
    fcopy (agreemntrec.origperson, code, 3);
    return TRUE;
}

/*
/*-----
referred_by : referred_by field
-----*/
int referred_by ()
{
    wintype win,win2;
    int key, iostat;
    char code[4];
        strcpy (code," ");
        win = windowopen (&entry_win);
        setttitle (win,"Referred By",CenterUpperTitle);
        cursoron ();
        key = get_line (code,20,1,3,win,"Referred By Code --> ");
        if (key == K_F2) {
            return FALSE;
        }
        iostat = 0;
        fcopy (rapersonrec.rapid, code, 3);
        if (!is_field_empty (code) )
            iostat = exactkey9 (fd_raperson, &rapersonrec);
        windowclose (win);
        if (iostat < 0)
            return FALSE;
        fcopy (agreemntrec.referredby, code, 3);
    return TRUE;
}
*/

/*-----
-----
Main_Window_Open : display Main operating window;
-----
*/
Main_Window_Open ()
{
    clrscr();
    main_wt = windowopen (&main_win);
    setttitle (main_wt,"* Renting Out A Phone
*",CenterUpperTitle);
}

```


OPENAGR.C

```

/*-----
list_window_open
-----*/
/
list_window_open () {
    list_wt = windowopen (&list_win);
    setttitle (list_wt,"Commands",CenterUpperTitle);
    gotoxy (1,1);
    cprintf ("F2  - Cancel");
    gotoxy (1,2);
    cprintf ("F3  - Finish");
    gotoxy (1,3);
    cprintf ("F9  - What Next?");
    gotoxy (1,4);
    cprintf ("F10 - More Options");
}

/*-----
init_fields_open: initialize fields open contract
-----*/
init_fields_open ()
{
    int i;
    agreemntrec = nulledrec;
    get_curdate (agreemntrec.rentaldate);    /* put rental date in
field */
    get_time (agreemntrec.timeout);
    get_curdate (agreemntrec.actrtndate);    /* so informix won't
choke */
    fcopy (agreemntrec.origagency,controlrec.tau_id,4);
    agreemntrec.phochgday = controlrec.phone_daily_chg;
    agreemntrec.phochgmin = controlrec.charge_per_minute;
    CARD_APPROVED = FALSE;
    agreemntrec.remarks5[0] = 'N'; /* this is now a flag for LDW */
}

/*-----
--
process_all: run all procesess
-----
-*/
process_all ()
{
    DATA_OPEN = FALSE;
    PRINTED_CONTRACT = FALSE;
    MANUAL = FALSE;
    RETRY_CREDIT = FALSE;
    if (!start_CTI_open()) return ;
    if (!start_credit_open()) return ;
    if (!start_data_open()) return ;
}

/*-----
--

```


OPENAGR.C

```

start_credit_open: show credit window and start by getting card swipe;
-----
-*/
start_credit_open ()
{
char ch,key;
wintype win;
int done = FALSE;

    credit_wt = windowopen (&credit_win);
    settitle (credit_wt,"Customer's Credit Card",CenterUpperTitle);
    cursoroff ();
    cprintf (" STEP 2 -> Swipe Credit Card OR Press F7 <-");
    while (!done)
        if (is_extended_key (ch = getch (),&key) && (key == K_F2)) {
            return FALSE;
        } else {
            if ((ch != '%') && (key != K_F7)){
                win = windowopen (&error_win);
                cprintf ("\n");
                settitle (win," CANCEL or SWIPE CARD
",CenterUpperTitle);
                centerPrint (60,"Only F2, F7 or Card Swipe are
allowed!");
                cprintf ("\n");
                centerPrint (60,"Press ANY Key Now");
                ch = getch();
                windowclose (win);
            } else done = TRUE;
            if (key == K_F7) done = TRUE;
        }

    if (key != K_F7){
        read_in_card(agreemntrec.creditno,
                    agreemntrec.custname,
                    agreemntrec.expireddate,
                    agreemntrec.creditttype);
        capAdjust (agreemntrec.custname,24);
        shorten_blanks (agreemntrec.custname);
        clrscr();
        cprintf (" Thank you...");
        credit_open ();
    } else {
        clrscr();
        cprintf (" Type in credit information and press F3");
        MANUAL = TRUE;
        credit_open();
    }
    return TRUE;
}

```

```

/*-----
-

```


OPENAGR.C

```
credit_open: Do credit authorization;
```

```
-----
```

```
*/
```

```
credit_open ()
```

```
{
```

```
int i,done;
```

```
char ch,key,s[80];
```

```
wintype win,win0,win1;
```

```
char response[80];
```

```
    if (MANUAL) return;    /* don't do credit if card was not swiped */
retry:
```

```
    done = FALSE;
```

```
    card_wt = windowopen (&card_win);
```

```
    settitle (card_wt,"Authorizing Card" ,CenterUpperTitle);
```

```
    gotoxy (1,3);
```

```
    strcpy (s," Card No: ");
```

```
    strcat (s,agreemntrec.creditno);
```

```
    cprintf ("%s",s);
```

```
    gotoxy (1,2);
```

```
    strcpy (s," Name: ");
```

```
    strcat (s,agreemntrec.custname);
```

```
    cprintf ("%s",s);
```

```
    gotoxy (1,4);
```

```
    strcpy (s," Expr: ");
```

```
    strcat (s,agreemntrec.expiredate);
```

```
    cprintf ("%s",s);
```

```
    CARD_APPROVED = get_credit (controlrec.preauth_amount,
```

```
                                credit_wt,
```

```
                                agreemntrec.creditno,
```

```
                                agreemntrec.expiredate,
```

```
                                controlrec.cdc_site_id,
```

```
                                controlrec.cdc_phone_number,
```

```
                                "33", /* Trans
```

```
code, 33, auth only */
```

```
                                response,
```

```
                                agreemntrec.preapproved,
```

```
                                1); /* 0=com1 1=com2 */
```

```
    use (card_wt);
```

```
    windowclose (card_wt);
```

```
    use (credit_wt); /* credit window */
```

```
    if (CARD_APPROVED) RETRY_CREDIT = FALSE;
```

```
    if (!CARD_APPROVED) {
```

```
        win0 = windowopen (&declined_win);
```

```
        settitle (win0," Credit Card Message
```

```
",CenterUpperTitle);
```

```
        gotoxy (1,1);
```

```
        centerPrint (60,response);
```

```
        gotoxy (1,3);
```

```
        centerPrint (60,"Press ESC to Exit or Swipe a
```


OPENAGR.C

Card!");

```

    while (!done) {
        ch = getch();
        if (ch == '%') {
            ungetch(ch); /* put back the % */
            read_in_card(agreemntrec.creditno,
                        agreemntrec.custname,
                        agreemntrec.expiredate,
                        agreemntrec.creditttype);
            capAdjust (agreemntrec.custname,24);
            shorten_blanks (agreemntrec.custname);
            textbackground (Black);

            if (DATA_OPEN) {
                display_card_name();
                display_card_number();
                display_card_expr();
            }

            done = TRUE;
            RETRY_CREDIT = TRUE;
        }
        if (ch == K_ESC) {
            done = TRUE;
            RETRY_CREDIT = FALSE;
        }
    }

    windowclose (win0);
    use (credit_wt);
}
if (RETRY_CREDIT) goto retry;
if (CARD_APPROVED) w_log_open (W_CREDIT); /* let what
next know done*/
if (CARD_APPROVED) agreemntrec.preauth_card[0] = 'Y';
}

```

```

/*-----
-
start_CTI_open
-----

```

```

-*/
int start_CTI_open ()
{
    static count = 0;
    char ch;

```

```

    CTI_wt = windowopen (&CTI_win);
    setttitle (CTI_wt," CTI Phone Process ",CenterUpperTitle);
    clrscr ();
    cprintf ("Step 1 -> Please Put Telephone in CTI Now!");
    cursoroff ();
    if (!CTI ())
        return FALSE;

```


OPENAGR.C

```

}

/*-----
---
CTI : do Phone initialization functions;
-----
--*/
int CTI ()
{
    int i, correct, stat;
    wintype win;
    char ch, key;
    char s[80];

    retry_phone:
        correct = FALSE;
        i = 0;

        stat = start_rtb ();
        if (stat <= 0) {
            errrtn ("Sorry - please start rental over");
            return FALSE;
        }
        w_log_open (W_MBC);
        return TRUE;
}

/*-----
-
start_data_open
-----
--*/
start_data_open ()
{
    data_wt = windowopen (&data_win);
    clrscr();
    setttitle (data_wt, " Data Entry Screen ", CenterUpperTitle);
    cursoron ();
    DATA_OPEN = TRUE;
    data ();
}

/*-----
display_scr1: Display fields for screen 1
-----*
/
display_scr1_open()
{
    use(data_wt);
    gotoxy (5, 2);
    cprintf ("Customer Name: ");

```


OPENAGR.C

```

gotoxy (5,3);
cprintf ("Card Number   : ");
gotoxy (5,4);
cprintf ("Expires         : ");
gotoxy (5,6);
cprintf ("Drivers License: ");
gotoxy (5,7);
cprintf ("Address   : ");
gotoxy (5,8);
cprintf ("City      : ");
gotoxy (5,9);
cprintf ("St/Zip    :   , ");
gotoxy (5,10);
        cprintf ("Home Phone : ");
        gotoxy (5,11);
        cprintf ("Local Phone: ");
        gotoxy (5,12);
        cprintf ("Return Date: ");
        gotoxy (39,3);
cprintf ("Portable Phone #: ");
gotoxy (39,4);
cprintf ("Agreement Number: ");

/* gotoxy (44,3);
   cprintf ("Meter Hours: ");
   gotoxy (44,4);
   cprintf ("Meter Mins : ");
*/
gotoxy (44,6);
cprintf ("Additional Equipment: ");
gotoxy (44,7);
cprintf ("-----");
gotoxy (44,8);
cprintf ("No. Extra Batteries: ");
gotoxy (44,9);
        cprintf ("No. Chargers           : ");
        gotoxy (44,10);
        cprintf ("LDW [Y/N]               : ");
gotoxy (44,11);
cprintf ("Discount %             : ");
}

/*-----
--
display_values_scr1() : display current values for screen 1
-----
--*/
display_values_scr1_open()
{
        display_card_name_open(data_wt);
        display_card_number_open(data_wt);
        display_card_expr_open(data_wt);
        display_phone_number_open(data_wt);
        /*

```


OPENAGR.C

```

        display_meter_hours_open(data_wt);
        display_meter_mins_open(data_wt);
        */
        display_drivers_open(data_wt);
        display_address_open(data_wt);
        display_city_open(data_wt);
        display_zip_open(data_wt);
        display_state_open(data_wt);
        display_home_phone_open(data_wt);
        display_local_phone_open (data_wt);
        display_estimated_return_date (data_wt);
/*      display_company_open(data_wt);  */
        display_batteries_open(data_wt);
        display_chargers_open(data_wt);
/*
        display_cases_open(data_wt);
*/
    display_ldw_open (data_wt);
    display_discount_open(data_wt);
    display_agreement_open(data_wt);
}

/*-----
final_checks_open() : check all fields for input
-----*/
/
int final_checks_open(int *field)
{
    if (is_field_empty (agreemntrec.custname) ) {
        *field = 1;
        strcpy (errmsg,"Customer Name Must Be Entered!");
        return FALSE;
    } else w_log_open (W_NAME);  /* else log as done */

        if (is_field_empty (agreemntrec.creditno) ) {
            strcpy (errmsg,"Credit Card Number Must Be Entered!");
            *field = 2;
            return FALSE;
        } else w_log_open (W_NUMBER);

    if (is_field_empty (agreemntrec.expiredate) ) {
        strcpy (errmsg,"Credit Card Expiration Date Must Be
Entered!");
        *field = 3;
        return FALSE;
    } else w_log_open (W_EXPR);

    if (is_field_empty (agreemntrec.licenseno) ) {
        strcpy (errmsg,"Driver's License Number Must Be Entered!
");
        *field = 4;
        return FALSE;
    } else w_log_open (W_DRIVERS);

```


OPENAGR.C

```
if (is_field_empty (agreemntrec.custaddr1) ) {
    strcpy (errmsg, "Customer's Home Address Must Be Entered!
");
    *field = 5;
    return FALSE;
} else w_log_open (W_ADDRESS);

if (is_field_empty (agreemntrec.custcity) ) {
    strcpy (errmsg, "Customer City Must Be Entered!");
    *field = 6;
    return FALSE;
} else w_log_open (W_CITY);

if (is_field_empty (agreemntrec.custstate) ) {
    strcpy (errmsg, "Customer's Home State Must Be Entered!");
;
    *field = 7;
    return FALSE;
} else w_log_open (W_ST);

if (is_field_empty (agreemntrec.custzipcd) ) {
    strcpy (errmsg, "Customer's Home Zip Code Must Be Entered
!");
    *field = 8;
    return FALSE;
} else w_log_open (W_ZIP);

    if (is_field_empty (agreemntrec.homephone) ) {
        strcpy (errmsg, "Customer's Home Phone Must Be Entered!");
;
        *field = 9;
        return FALSE;
    } else w_log_open (W_HOME_PHONE);

    if (is_field_empty (agreemntrec.local_phone_number) ) {
        strcpy (errmsg, "Customer's Local Phone Number
Must Be Entered!");
        *field = 10;
        return FALSE;
    } else w_log_open (W_LOCAL_PHONE);

    if (is_field_empty (agreemntrec.estimated_return_date) ) {
        strcpy (errmsg, "An Expected Rental Return Date
Must Be Entered!");
        *field = 11;
        return FALSE;
    } else w_log_open (W_ESTIMATED_RETURN_DATE);

    if (!w_is_logged_open (W_LDW)) {
        strcpy (errmsg, "You must ask customer if they want
LDW!");
        *field = 14;
        return FALSE;
```


OPENAGR.C

```

    }

    return TRUE;
}

/*-----
// Function Name -> do_open_F2
// Parameters:
// Function:
// Returns:
// Written By : Greg McGregor
//
-----*/
void do_open_F2 ( int *done ) {
    wintype win;
    if ( (!PRINTED_CONTRACT) && (!CARD_APPROVED) ){
        win = windowopen (&error_win);
        setttitle (win," F2 - CANCEL! ",CenterUpperTitle);
        gotoxy (5,2);
        if (yes_no ("Contract will be LOST, Are you sure
(Y/N)?",FALSE)) {
            centerPrint (60,"Wait A Minute While I Shut
Everything Down!");
            *done = TRUE;
        }
        windowclose (win);
    } else {
        errrtn("You Can't Cancel Now!");
    }
}

/*-----
// Function Name -> do_open_F3
// Parameters:
// Function:
// Returns:
// Written By : Greg McGregor
//
-----*/
void do_open_F3 (int *FIELD) {
    wintype win,note_win2,note_win;
    char errmessage[80];

    if ( (!is_field_empty (agreemntrec.creditno)) &&
        (!is_field_empty (agreemntrec.expiredate))) {
        MANUAL = FALSE; /* set global flag to automatic */
        RETRY_CREDIT = TRUE; /* retry or plain try if F3 */
        if (!CARD_APPROVED) {
            note_win2 = note ("Wait While Credit
Authorization is Completed!");
            credit_open();
            use (note_win2);

```


OPENAGR.C

```

        windowclose (note_win2);
    }
    if ( (CARD_APPROVED) && (!PRINTED_CONTRACT) ) {
        prt_error_number = -10;
        strcpy (prt_error_message, "Couldn't Finish Phone
Initialization!");
        if (final_checks_open (FIELD)) {
            add_upd_agreemnt (1); /* open contract = 1
*/
            print_contract (1, FALSE); /* opening
agreement */
        } else errrtn(errmessage);
        if (prt_error_number != 0){
            strcpy (errmessage, prt_error_message);
            errrtn (errmessage);
        } else {
            if (prt_error_number != -10) {
                note_win = note ("Wait One
Moment!");
                unlock_turn_off_phone ();
                use (note_win);
                windowclose (note_win);
                use (CTI_wt);
                clrscr ();
                centerPrint (50, "  -* Remove Phone
From CTI *-  ");
                use (data_wt);
                w_log_open (W_PRINTED); /* log
successful print*/
                PRINTED_CONTRACT = TRUE;
            }
        }
    }
    use (credit_wt);
} else {
    win = windowopen (&error_win);
    setttitle (win, " A Problem Has occured! ", CenterUpperTitle);
    gotoxy (1, 2);
    centerPrint (60, "Please enter credit card information.");
    gotoxy (1, 3);
    centerPrint (60, "Press ANY key to continue");
    getch();
    windowclose (win);
    use (credit_wt);
}
use (data_wt);
}

```

```

/*-----
// Function Name -> do_open_F5
// Parameters:

```


OPENAGR.C

```

// Function:
// Returns:
// Written By : Greg McGregor
//
-----*/
void do_open_F5 (int *FIELD ) {
char errmessage[80];
wintype note_win;

    if (!CARD_APPROVED) {
        errrtn ("Credit Card Approval Has NOT Been Completed!");
    } else
    if (final_checks_open (FIELD)) {
        add_upd_agreemnt (1); /* open contract = 1 */
        print_contract (1,FALSE);
    } else errrtn(errmessage);
    if ( (prt_error_number != 0) && (CARD_APPROVED) ){
        strcpy (errmessage,prt_error_message);
        errrtn (errmessage);
    } else {
        note_win = note ("Wait One Moment!");
        unlock_turn_off_phone ();
        use (note_win);
        windowclose (note_win);
        use (CTI_wt);
        clrscr ();
        centerPrint (50,"  -* Remove Phone From CTI *-  ");
        use (data_wt);
        w_log_open (W_PRINTED); /* log successful print*/
        PRINTED_CONTRACT = TRUE;
    }
    use (data_wt);
}

/*-----*/
// Function Name -> do_open_F6
// Parameters:
// Function:
// Returns:
// Written By : Greg McGregor
//
-----*/
void do_open_F6 ( int *done , int *win_open) {
wintype win;
    if (!CARD_APPROVED) {
        *win_open = TRUE;
        win = windowopen (&error_win);
        setttitle (win,"ERROR",CenterUpperTitle);
        gotoxy (1,1);
        centerPrint (60,"Credit Card Approval Not Completed!");
    }
    if (!PRINTED_CONTRACT) {
        if (!*win_open) {

```


OPENAGR.C

```

        win = windowopen (&error_win);
        setttitle (win,"ERROR",CenterUpperTitle);
    }
    gotoxy (1,3);
    centerPrint (60,"Opening Agreement Has NOT Been Printed!");
}
if ( (CARD_APPROVED) && (PRINTED_CONTRACT) ) {
    *done = TRUE;
    update_tau_status (0,'1');
    add_upd_agreemnt (1); /* open contract = 1 */
    system ("ccopyit agreemnt. ");
    system ("ccopyit phone.");
} else {
    *win_open = FALSE;
    gotoxy (1,4);
    centerPrint(60,"Press ANY key to Exit");
    ahoh ();
    getch();
    windowclose (win);
}
use (data_wt);
}

/*-----
// Function Name -> do_open_F7
// Parameters:
// Function:
// Returns:
// Written By : Greg McGregor
//
-----*/
void do_open_F7 () {
    wintype note_win2,win;
    char errmessage[80];

    if (CARD_APPROVED) {
        strcpy (errmessage,"Credit Card Already Authorized!");
        errrtn (errmessage);
    } else
    if ( (!is_field_empty (agreemntrec.creditno)) &&
        (!is_field_empty (agreemntrec.expiredate))) {
        MANUAL = FALSE; /* set global flag to automatic */
        RETRY_CREDIT = TRUE; /* retry or plain try if F3 */
        if (!CARD_APPROVED) {
            note_win2 = note ("Wait While Credit
Authorization is Completed!");
            credit_open();
            use (note_win2);
            windowclose (note_win2);
        }
        use (credit_wt);
    } else {
        win = windowopen (&error_win);

```


OPENAGR.C

```

    settitle (win, " A Problem Has occurred! ", CenterUpperTit
le);

    gotoxy (1,2);
    centerPrint (60, "Please enter credit card information.");
    gotoxy (1,3);
    centerPrint (60, "Press ANY key to continue");
    getch();
    windowclose (win);
    use (credit_wt);
}
use (data_wt);
}

/*-----
// Function Name -> do_open_F8
// Parameters:
// Function:
// Returns:
// Written By : Greg McGregor
//
-----*/

void do_open_F8 ( ) {
wintype man_win;
char errmessage[80];
char temp[80];

    if (CARD_APPROVED) {
        strcpy (errmessage, "Credit Authorization Already
Completed!");
        errrtn(errmessage);
    } else {
        man_win = windowopen (&manual_win);
        settitle (man_win, "Authorized By Rental
Agent", CenterUpperTitle);
        gotoxy (5,1);
        cprintf ("Authorization Number -> ");
        temp[0] = '\0';
        get_line (temp, 5, 1, 6, man_win, "Authorization Number -> ");

        moveX (agreemntrec.preapproved, temp, 6);
        CARD_APPROVED = TRUE; /* done and approved */
        w_log_open (W_CREDIT); /* log credit */
        windowclose (man_win);
        use (credit_wt);
        clrscr();
        cprintf("          Credit Authorization Number : %s", temp);
    }
    use (data_wt);
}

/*-----

```


OPENAGR.C

```

// Function Name -> do_open_F9
// Parameters:
// Function:
// Returns:
// Written By : Greg McGregor
//
-----*/
void do_open_F9 ( int *FIELD ) {
    final_checks_open(&FIELD);    /* check required fields */
    what_next_open ();
    use (data_wt);
}

/*-----
-
Data :  get data entry;
-----
-*/
data ()
{
    char s[80];
    int win_open = FALSE;
    int done_window = FALSE;
    int done = FALSE;
    int lastStop = FALSE;
    wintype win,win0,win1,man_win,note_win,note_win2;
    char temp[10];
    int FIELD = 4;
    int try,key;
    cti_obj sco;    /* starting Cti object */

    try = 1;
    use (data_wt);
    display_scr1_open();
    display_values_scr1_open();
    /* start at Credit info if manual type in */
    if (MANUAL) FIELD = 1;
    while (!done ){
        switch (FIELD) {
            case 1: key = get_card_name_open (data_wt);
                    break;
            case 2: key = get_card_number_open (data_wt);
                    break;
            case 3: key = get_card_expr_open (data_wt);
                    break;
            case 4: key = get_drivers_open (data_wt);
                    break;
            case 5: key = get_address_open (data_wt);
                    break;
            case 6: key = get_city_open(data_wt);
                    break;
            case 7: key = get_state_open(data_wt);
                    break;

```


OPENAGR.C

```

        case 8: key = get_zip_open(data_wt);
                break;
        case 9: key = get_home_phone_open(data_wt);
                break;
        case 10: key = get_local_phone_open (data_wt);
                break;
        case 11: key = get_estimated_return_date (data_wt);
                break;
        case 12: key = get_batteries_open(data_wt);
                break;
        case 13: key = get_chargers_open(data_wt);
                break;
        case 14: key = get_ldw_open (data_wt);
                w_log_open (W_LDW);
                break;
        case 15: key = get_discount_open(data_wt);
                break;
    } /* switch end */
    if (key == K_F1) {
        help_list_open ();
        use (data_wt);
    }
    if (key == K_F10) {
        command_list_open ();
        use (data_wt);
    }
    if (UP_FIELD) {
        if (FIELD > 1){ --FIELD; }
        else if (FIELD == 1) FIELD = 15;
    }
    if (DOWN_FIELD) {
        if (FIELD < 15) { ++FIELD; }
        else if ( FIELD == 15 ) FIELD = 1;
    }

    if ( key == K_F2 ) do_open_F2 ( &done );

    if ( key == FORCED_EXIT ) done = TRUE;

    if ( key == K_F3 ) do_open_F3 ( &FIELD );

    if ( key == K_F5 ) do_open_F5 ( &FIELD );

    if ( key == K_F6 ) do_open_F6 ( &done , &win_open);

    if ( key == K_F7 ) do_open_F7 ( );

    if ( key == K_F8 ) do_open_F8 ( );

    if ( key == K_F9 ) do_open_F9 ( &FIELD );

```

}

}

OPENAGR.C

```

/*-----
---
command_list_open: show command list
-----
--*/
command_list_open ()
{
wintype win;
char c;

    win = windowopen (&commands_win);
    settitle (win," Commands List ",CenterUpperTitle);
    gotoxy (1,1);
    cprintf ("      F1  - Help");
    gotoxy (1,2);
    cprintf ("      F2  - Cancel, 'Get Me Out Key'");
    gotoxy (1,3);
    cprintf ("      F3  - Finish Key");
    gotoxy (1,4);
    cprintf ("      F4  - ");
    gotoxy (1,5);
    cprintf ("      F5  - Print Receipt");
    gotoxy (1,6);
    cprintf ("      F6  - Exit, 'I am all done!'");
    gotoxy (1,7);
    cprintf ("      F7  - Retry Credit Authorization");
    gotoxy (1,8);
    cprintf ("      F8  - Authorized By Rental Agent");
    gotoxy (1,9);
    cprintf ("      F9  - What Do I Do Next (?) Key");
    gotoxy (1,10);
    cprintf ("                      ESC - EXIT ");
    while ((c = getch ()) != K_ESC) ;
    windowclose (win);
}

/*-----
---
help_list_open: show command list
-----
--*/
help_list_open ()
{
wintype win;
char c;

    win = windowopen (&commands_win);
    settitle (win," Quick Step Help ",CenterUpperTitle);
    gotoxy (1,1);
    cprintf ("                      STEP");
    gotoxy (1,2);
    cprintf ("                      ----");
    gotoxy (1,3);
    cprintf ("      1  - Put Phone in MBC Box");

```


OPENAGR.C

```
gotoxy (1,4);
cprintf ("      2 - Swipe Credit Card");
gotoxy (1,5);
cprintf ("      3 - Type Name, Address, Etc..");
gotoxy (1,6);
cprintf ("      4 - Press F3 to Finish");
gotoxy (1,8);
cprintf ("      You Are All Done!");
gotoxy (1,9);
cprintf ("      ESC - EXIT ");
while ((c = getch ()) != K_ESC) ;
windowclose (win);
}
```


MS-DOS MACRO ASSEMBLER A51 V4.4
OBJECT MODULE PLACED IN CTI_BEGN.OBJ
ASSEMBLER INVOKED BY: A51 CTI_BEGN.A51 DEBUG ERRORPRINT(CTI_BEGN.ERR)
NOSYMBOLS NOXREF

LOC	OBJ	LINE	SOURCE
		1	\$PAGEWIDTH (127)
		2	\$PAGELENGTH (57)
		3	;
		4	\$TITLE (CTI_BEGN.A51)
		5	;
		6	;
		7	Program Title: Cellular Telephone Interface Controller Firmwa
		8	Filename : CTI_BEGN.A51
		9	Module Name : CTI_BEGN.OBJ
		10	Project # :
		11	Author : Theodore W. Watler
		12	From : Parchment Designs
		13	For : Turner, Gold, France & Associates
		14	Date Created : August 2, 1991
		15	Version : A.00
		16	;
		17	;
		18	COPYRIGHT (C) 1991. ALL RIGHTS RESERVED
		19	Turner, Gold, France & Associates
		20	;
		21	;
		22	;
		23	PROGRAM FUNCTION
		24	;
		25	;
		26	PROGRAM DESCRIPTION
		27	;
		28	;
		29	REFERENCES
		30	;
		31	1. 8051 Hardware Reference Manual
		32	2. Franklin Software DK51 Development Tools
		33	3.
		34	;
		35	***** MODULE HISTORY *****
		36	;
		37	;
		38	#####

LOC	OBJ	LINE	SOURCE
		39	\$EJECT
		40	;
		41	NAME CTI_FIRMWARE_START
		42	;
		43	EXTERNAL REFERENCE TABLE
		44	;
		45	EXTRN CODE (CTPHONE_ACTIVE_ISR)
		46	EXTRN CODE (CTI_TIMEOUT_ISR)
		47	EXTRN CODE (HOST_XFER_ISR)
		48	EXTRN CODE (TEST_ROM_CHECKSUM)
		49	EXTRN CODE (CTI_MAIN_FUNCTION)
		50	EXTRN CODE (TIME_DELAY)
		51	;
		52	PUBLIC DECLARATION TABLE
		53	;
		54	PUBLIC FAILED_SELFTEST
		55	PUBLIC STACK
		56	;
		263	\$LIST
		264	;
		265	
		266	CTI_BEGIN SEGMENT CODE
		267	STACK_SEG SEGMENT IDATA
		268	;
----		269	RSEG STACK_SEG
0000		270	STACK: DS 020H ; 32 Byte Deep Stack
		271	

LOC	OBJ	LINE	SOURCE
		272	\$EJECT
		273	;
		274	;
		275	;
		276	CSEG AT 0
0000	020000	F 277	ljmp POWER_ON_RESET ; Power on reset vector
		278	;
		279	;
		280	;
0003		281	ORG 0003H
0003	020000	F 282	ljmp CTPHONE_ACTIVE_ISR ; ~INT0, PTR-800 Input
		283	;
		284	;
		285	;
000B		286	ORG 000BH
000B	020000	F 287	ljmp CTI_TIMEOUT_ISR ; T0, CTI data xfers timeout
		288	;
		289	;
		290	;
0013		291	ORG 0013H ; ~INT1, Unused
0013	32	292	reti
		293	;
		294	;
		295	;
001B		296	ORG 001BH ; T1, Unused
001B	32	297	reti
		298	;
		299	;
		300	;
0023		301	ORG 0023H ; Serial Receive/Xmit Interrupt
0023	020000	F 302	ljmp HOST_XFER_ISR
		303	;
		304	;
		305	;
002B		306	ORG 002BH ; T2, Not Used (8052 ONLY)
002B	32	307	reti
		308	

LOC	OBJ	LINE	SOURCE
		309	\$EJECT
		310	;
0033		311	ORG 0033H
----		312	RSEG CTI_BEGIN
		313	USING REG_BANK_00
		314	;
		315	POWER-ON RESET ENTRY
		316	;
0000		317	POWER_ON_RESET:
		318	;
		319	Configure the I/O ports P1 & P3
		320	;
0000	7590E8	321	mov P1, #11101000B ; Configure port P1 I/O
0003	75B07F	322	mov P3, #01111111B ; Configure port P3 I/O
		323	;
		324	Assure the system is completely disabled
		325	;
0006	E4	326	clr a ; Clear ACC cuz we need a zero
0007	F5A8	327	mov ie, a ; Disable all interrupts
0009	F5D0	328	mov psw, a ; Select Register bank zero
000B	F588	329	mov tcon, a ; Disable all the 8031 timers
000D	758100 F	330	mov sp, #low STACK-1 ; Initialize the top of stack
		331	;
		332	Test the 8031 registers after RESET
		333	;
0010	45F0	334	orl a, b ; Test B register
0012	4582	335	orl a, dpl ; Test DPL register
0014	4583	336	orl a, dph ; Test DPH register
0016	45D0	337	orl a, psw ; Test PSW register
0018	4598	338	orl a, scon ; Test SCON register
001A	458A	339	orl a, tl0 ; Test TL0 register
001C	458C	340	orl a, th0 ; Test TH0 register
001E	458B	341	orl a, tl1 ; Test TL1 register
0020	458D	342	orl a, th1 ; Test TH1 register
0022	4588	343	orl a, tcon ; Test TCON register
0024	4589	344	orl a, tmod ; Test TMOD register
0026	6005	345	jz INTERNAL_RAM_TEST ; Continue The Selftest
		346	;
0028	7400	347	mov a, #M8031_FAULT ; Failed micro selftest
002A	020000 F	348	ljmp FAILED_SELFTEST ; Failed Registers Test
		349	;
		350	

LOC OBJ LINE SOURCE

```

351          $EJECT
352 ;
353 ;   Test the 8031 Internal RAM (All 128 Bytes)
354 ;
002D          355 INTERNAL_RAM_TEST:
002D F8          356          mov  r0, a          ; Internal Ram Start Address
002E F4          357          cpl  a          ; Start Pattern 0FFH
358 ;
002F          359 RAM_TEST00:
002F F6          360          mov  @r0, a          ; Move test pattern to address
0030 04          361          inc  a          ; Generate next test pattern
0031 08          362          inc  r0          ; Generate next RAM address
0032 B880FA      363          cjne  r0, #80H, RAM_TEST00 ; Repeat if not last address
364 ;
365 ;   Verify the test written data correctness
366 ;
0035 14          367          dec  a          ; Restore the test pattern
0036 18          368          dec  r0          ; Restore last RAM address
369 ;
0037          370 RAM_TEST01:
0037 66          371          xrl  a, @r0          ; Compare the test data
0038 6005        372          jz   RAM_TEST02      ; Continue the RAM Test
373 ;
003A 7400        374          mov  a, #M8031_FAULT ; Failed micro selftest
003C 020000 F    375          ljmp  FAILED_SELFTEST ; Failed Internal RAM Test
376 ;
003F          377 RAM_TEST02:
003F 66          378          xrl  a, @r0          ; Restore the test pattern
0040 14          379          dec  a          ; Generate the next test pattern
0041 D8F4        380          djnz  r0, RAM_TEST01 ; Check all 128 byte locations
0043 B400E9      381          cjne  a, #00H, RAM_TEST00 ; Test All Possible test
patterns
382 ;
383

```


LOC OBJ LINE SOURCE

```

384          $EJECT
385          ;
386          ;   Test the 8031 Register Banks
387          ;
0046          388 REGISTER_BANK_TEST:
0046 75F008    389          mov  b, #08H          ; Number of registers per bank
390          ;
0049          391 REGISTER_BANK00:
0049 66        392          xrl  a, @r0          ; Compare ACC with a register
004A 6005      393          jz   REGISTER_BANK01  ; Continue the register bank
test
394          ;
004C 7400      395          mov  a, #M8031_FAULT    ; Failed micro selftest
004E 020000 F 396          ljmp  FAILED_SELFTEST    ; Failed register bank test
397          ;
0051          398 REGISTER_BANK01:
0051 66        399          xrl  a, @r0          ; Restore test pattern
0052 04        400          inc  a              ; Generate next test pattern
0053 08        401          inc  r0             ; Point to the next register
0054 D5F0F2    402          djnz  b, REGISTER_BANK00 ; Check all 8 registers
403
0057 F5F0      404          mov  b, a              ; Save the test data pattern
0059 E5D0      405          mov  a, psw          ; Get the current PSW
005B 2408      406          add  a, #08H        ; Select the next register bank
005D F5D0      407          mov  psw, a         ; Restore the updated PSW
005F E5F0      408          mov  a, b          ; Restore the current test data
0061 B420E2    409          cjne  a, #20H, REGISTER_BANK_TEST
0064 75D000    410          mov  psw, #00H      ; Reset the PSW
411          ;
412

```


LOC OBJ LINE SOURCE

```

413          $EJECT
414          ;
415          ;   Test the 8031 timer registers
416          ;
0067          417 TIMERS_TEST:
0067 758CFF    418          mov  th0, #0FFH          ; Setup the timer 0 for a count
006A 758AF0    419          mov  tl0, #0F0H          ; of 16 machine cycles
006D 758DFF    420          mov  th1, #0FFH          ; Setup the timer 1 for a count
0070 758BF0    421          mov  tl1, #0F0H          ; of 16 machine cycles
0073 758850    422          mov  tcon, #50H          ; Enable both timers
0076 D5E0FD    423          djnz  acc, $              ; Delay Loop, ACC = 20H
0079 308D05    424          jnb  tf0, FAILED_TIMERS_TEST ; Failed Timer Test
007C 308F02    425          jnb  tf1, FAILED_TIMERS_TEST ; Failed Timer Test
007F 8005      426          sjmp  CLEAR_TIMERS        ; Clear timers & continue
selftest
427          ;
0081          428 FAILED_TIMERS_TEST:
0081 7400      429          mov  a, #M8031_FAULT      ; Failed micro selftest
0083 020000 F 430          ljmp  FAILED_SELFTEST      ; Failed the processor
selftest
431          ;
0086          432 CLEAR_TIMERS:
0086 E4        433          clr  a                    ; Clear the accumulator
0087 F5D0      434          mov  psw, a                ; Reset the PSW
0089 F588      435          mov  tcon, a              ; Disable the timers
008B F58C      436          mov  th0, a                ; Clear TH0 register
008D F58A      437          mov  tl0, a                ; Clear TL0 register
008F F58D      438          mov  th1, a                ; Clear TH1 register
0091 F58B      439          mov  tl1, a                ; Clear TL1 register
0093 8000      440          sjmp  CLEAN_UP_RAM
441          ;
442          ;   Test the Firmware ROM Checksum
443          ;
444          ;   lcall TEST_ROM_CHECKSUM
445          ;   jc  CLEAN_UP_RAM
446          ;
447          ;   mov  a, #PROM_FAULT                  ; Failed PROM Test
448          ;   ljmp  FAILED_SELFTEST                ; Failed the processor selftest
449

```


LOC	OBJ	LINE	SOURCE
		450	\$EJECT
		451	;
		452	; Clear the internal RAM. Thus the program begins with the RAM Initialize
		453	;
0095		454	CLEAN_UP_RAM:
0095 E4		455	clr a ; Clear ACC cuz we need a zero
0096 F8		456	mov r0, a ; Clear R0 register
		457	;
0097		458	CLEAR_INTERNAL_RAM:
0097 F6		459	mov @r0, a ; Clear RAM location
0098 08		460	inc r0 ; Bump address pointer to next
0099 B880FB		461	cjne r0, #80H, CLEAR_INTERNAL_RAM
009C F8		462	mov r0, a ; Clear R0 register
		463	

LOC OBJ LINE SOURCE

```

464          $EJECT
465          ;
466          ; Setup the beginning state of the control register
467          ;
009D 75A880 468      mov  ie, #10000000B      ; Set all interrupts flags
00A0 75B802 469      mov  ip, #00000010B      ; Set timeout timer with the
highest level
00A3 758DFD 470      mov  th1, #BAUD_9600      ; Setup for 9600 baud
transmission
00A6 758BFD 471      mov  tl1, #BAUD_9600      ; Setup for 9600 baud
transmission
00A9 758C00 472      mov  th0, #00H            ; Clear TH0
00AC 758A00 473      mov  tl0, #00H            ; Clear TL0
00AF 758700 474      mov  pcon, #00000000B      ; Set SMOD for no baud
doubling
00B2 759850 475      mov  scon, #01010000B      ; Set serial control register
00B5 758841 476      mov  tcon, #01000001B      ; Set timer control to all bits off
00B8 758921 477      mov  tmod, #00100001B      ; Select T1 Mode 2, T0 Mode 1
478          ;
00BB C200 F 479      clr  CTI_LED_1            ; If phone ignore it
00BD D200 F 480      setb CTI_LED_2            ; Flag the CTI is alive
00BF C200 F 481      clr  CTI_PTR_EXPWR_ON      ; Assure the phone is off
482          ;
483          ; Go startup the CTI functions and begin the formal program
484          ;
00C1 020000 F 485      ljmp CTI_MAIN_FUNCTION      ; Go start that main
program
486          ;
00C4          487  FAILED_SELFTEST:
00C4 C2AF      488      clr  ea                ; Turn off global Interrupts
00C6 120000 F 489      call FLASH_ERROR_LED_2      ; Show the error code on
the LED
00C9 80F9      490      sjmp FAILED_SELFTEST      ; Crashed wait for a reset
491

```


LOC	OBJ	LINE	SOURCE
		492	\$EJECT
		493	;
		494	;*<
		495	; NAME:
		496	;
		497	; DESCRIPTION:
		498	; CALL:
		499	; ARGUMENTS:
		500	; MODIFIES:
		501	; RETURNS:
		502	; HISTORY:
		503	;*>
		504	;
00CB		505	FLASH_ERROR_LED_2:
		506	;
00CB	120000	F 507	lcall DISPLAY_ONE_BIT
00CE	120000	F 508	lcall DISPLAY_ZERO_BIT
00D1	22	509	ret
		510	

LOC OBJ LINE SOURCE

```

511          $EJECT
512          ;
513          ;*<
514          ; NAME:
515          ;
516          ; DESCRIPTION:
517          ; CALL:
518          ; ARGUMENTS:
519          ; MODIFIES:
520          ; RETURNS:
521          ; HISTORY:
522          ;*>
523          ;
00D2          524 DISPLAY_ONE_BIT:
525          ;
00D2 D200 F 526          setb CTI_LED_2          ; Turn on the activity LED
527          ;
00D4 750005 F 528          mov  DELAY_CTR_02, #005H          ; Setup Delay MSB
00D7 750038 F 529          mov  DELAY_CTR_01, #038H          ; Setpu Delay Middle
byte
00DA 120000 F 530          lcall TIME_DELAY          ; Delay for 562 milliseconds
531          ;
00DD C200 F 532          clr  CTI_LED_2          ; Turn off the activity LED
533          ;
00DF 750002 F 534          mov  DELAY_CTR_02, #002H          ; Setup Delay MSB
00E2 750068 F 535          mov  DELAY_CTR_01, #068H          ; Setpu Delay Middle
byte
00E5 120000 F 536          lcall TIME_DELAY          ; Delay for 188 milliseconds
537          ;
00E8 22          538          ret
539

```


LOC	OBJ	LINE	SOURCE
		540	\$EJECT
		541	;
		542	;*<
		543	; NAME: Display Zero Bit
		544	;
		545	; DESCRIPTION:
		546	; CALL:
		547	; ARGUMENTS:
		548	; MODIFIES:
		549	; RETURNS:
		550	; HISTORY:
		551	;*>
		552	;
00E9		553	DISPLAY_ZERO_BIT:
		554	;
00E9 D200	F	555	setb CTI_LED_2 ; Turn on the activity LED
		556	;
00EB 750002	F	557	mov DELAY_CTR_02, #002H ; Setup Delay MSB
00EE 750068	F	558	mov DELAY_CTR_01, #068H ; Setpu Delay Middle
			byte
00F1 120000	F	559	lcall TIME_DELAY ; Delay for 188 milliseconds
		560	;
00F4 C200	F	561	clr CTI_LED_2 ; Turn off the activity LED
		562	;
00F6 750005	F	563	mov DELAY_CTR_02, #005H ; Setup Delay MSB
00F9 750038	F	564	mov DELAY_CTR_01, #038H ; Setpu Delay Middle
			byte
00FC 120000	F	565	lcall TIME_DELAY ; Delay for 582 milliseconds
		566	;
00FF 22		567	ret
		568	
		569	END
			;
			; End of CTI_BEGN.A51
			;

REGISTER BANK(S) USED: 0

ASSEMBLY COMPLETE, NO ERRORS FOUND

MS-DOS MACRO ASSEMBLER A51 V4.4
OBJECT MODULE PLACED IN CTI_CMDS.OBJ
ASSEMBLER INVOKED BY: A51 CTI_CMDS.A51 DEBUG ERRORPRINT(CTI_CMDS.ERR)
NOSYMBOLS NOXREF

LOC	OBJ	LINE	SOURCE
		1	\$PAGEWIDTH (127)
		2	\$PAGELENGTH (57)
		3	;
		4	\$TITLE (CTI_CMDS.A51)
		5	;
		6	;
		7	Program Title: Cellular Telephone Interface Controller Firmwa
		8	Filename : CTI_CMDS.A51
		9	Module Name : CTI_CMDS.OBJ
		10	Project # :
		11	Author : Theodore W. Watler
		12	From : Parchment Designs
		13	For : Turner, Gold, France & Associates
		14	Date Created : August 4, 1991
		15	Version : A.00
		16	;
		17	;
		18	;
		19	COPYRIGHT (C) 1991. ALL RIGHTS RESERVED
		20	Turner, Gold, France & Associates
		21	;
		22	;
		23	;
		24	PROGRAM FUNCTION
		25	;
		26	;
		27	PROGRAM DESCRIPTION
		28	;
		29	;
		30	REFERENCES
		31	;
		32	1. 8051 Hardware Reference Manual
		33	2. Franklin Software DK51 Development Tools
		34	3.
		35	;
		36	***** MODULE HISTORY *****
		37	;
		38	;
			#####

LOC	OBJ	LINE	SOURCE
		39	\$EJECT
		40	;
		41	NAME CTI_COMMAND_PROCESSOR
		42	;
		43	EXTERNAL REFERENCE TABLE
		44	;
		45	EXTRN CODE (RECEIVE_HOST_DATA)
		46	EXTRN CODE (TRANSFER_HOST_DATA)
		47	
		48	EXTRN CODE (RECEIVE_PHONE_DATA)
		49	EXTRN CODE (TRANSFER_PHONE_DATA)
		50	
		51	EXTRN CODE (DEC_HOST_XFER_COUNT)
		52	EXTRN CODE (CHECK_PHONE_STATUS)
		53	EXTRN CODE (SETUP_HOST_TIMEOUT)
		54	EXTRN CODE (SETUP_TPHONE_TIMEOUT)
		55	EXTRN CODE (TIME_DELAY)
		56	EXTRN CODE (DELAY_350_MSECS)
		57	EXTRN CODE (DELAY_WRITE_RAM_FIRMWARE)
		58	;
		59	PUBLIC DECLARATION TABLE
		60	;
		61	PUBLIC CMD_READ_PTR_PHONE_NUMBER
		62	PUBLIC CMD_READ_PHONE_CALLS
		63	PUBLIC CMD_READ_PHONE_TIME
		64	PUBLIC CMD_WRITE_PHONE_TIME
		65	PUBLIC CMD_TURN_PHONE_OFF
		66	PUBLIC CMD_READ_PHONE_RTb_VER
		67	PUBLIC CMD_READ_NOVATEL_VER
		68	PUBLIC CMD_READ_CTI_VERSION
		69	PUBLIC CMD_TURN_POWER_ON
		70	PUBLIC CMD_LOCK_PHONE
		71	PUBLIC CMD_UNLOCK_PHONE
		72	PUBLIC CMD_READ_AIR_TIME_METER
		73	PUBLIC CMD_FAKE_POWER_DOWN
		74	PUBLIC CMD_READ_CALLS_COUNTER
		75	PUBLIC CMD_READ_CALLS_RAM_PTR
		76	PUBLIC CMD_WRITE_TELEMAC_FIRMWARE
		77	PUBLIC CMD_PHONE_IN_CRADDLE
		78	PUBLIC CMD_RESET_CALLS_POINTER
		79	PUBLIC CMD_RESET_CALLS_COUNTER
		80	PUBLIC CMD_RESET_AIR_TIME_METER
		81	PUBLIC CTI_FIRMWARE_REVISION
		82	;
		289	\$LIST
		290	;
		291	CTI_COMMANDS SEGMENT CODE
----		292	RSEG CTI_COMMANDS
		293	USING REG_BANK_00
		294	

LOC	OBJ	LINE	SOURCE
		295	\$EJECT
		296	;
		297	;*<
		298	NAME:
		299	;
		300	DESCRIPTION:
		301	CALL:
		302	ARGUMENTS:
		303	MODIFIES:
		304	RETURNS:
		305	HISTORY:
		306	;*>
		307	;
0000		308	CMD_READ_PTR_PHONE_NUMBER:
0000		309	CMD_READ_PHONE_TIME:
0000		310	CMD_READ_PHONE_RTb_VER:
0000		311	CMD_READ_NOVATEL_VER:
0000		312	CMD_READ_AIR_TIME_METER:
0000		313	CMD_READ_CALLS_COUNTER:
0000		314	CMD_READ_CALLS_RAM_PTR:
		315	;
0000 D200	F	316	setb CTI_HOST_CTS ; Flag host not ready for action!!!
0002 120000	F	317	call CHECK_PHONE_STATUS ; If the phone awake ???
0005 300021	F	318	jnb LIVE_PHONE_IN_CRADDLE, CMD_READ_PTR_EXIT
0008 30001E	F	319	jnb HOST_TURN_PHONE_ON, CMD_READ_PTR_EXIT
		320	;
000B E500	F	321	mov a, HOST_CMD_REG
000D 120000	F	322	call TRANSFER_PHONE_DATA ; Xfer the CTPHONE command
0010 200016	F	323	jb PTR_XFER_TIMEDOUT, CMD_READ_PTR_EXIT
		324	;
0013 750000	F	325	mov PTR_CHECKSUM_REG, #00H ; Clear CTPHONE data checksum register
0016 AB00	F	326	mov r3, CTI_BUFFER_COUNT ; CTPHONE # of return data bytes
0018 7800	F	327	mov r0, #low XFER_DATA_BUFFER ; Address for the data buffer
		328	;
		329	Read the CTPHONE returned data
		330	;
001A		331	CMD_READ_PTR11:
001A 120000	F	332	call RECEIVE_PHONE_DATA ; A byte at a time
001D 200009	F	333	jb PTR_XFER_TIMEDOUT, CMD_READ_PTR_EXIT
		334	;
0020 F6		335	mov @r0, a ; If not timed out store byte
0021 08		336	inc r0 ; Point to next storage location
0022 6200	F	337	xrl PTR_CHECKSUM_REG, a ; Compute the checksum, as of old design
0024 DBF4		338	djnz r3, CMD_READ_PTR11
		339	;
		340	Transfer the CTPHONE returned data to the host
		341	;
0026 120000	F	342	call TRANSFER_HOST_DATA


```
0029      343 ;  
      344 CMD_READ_PTR_EXIT:  
0029 22    345      ret
```


LOC	OBJ	LINE	SOURCE
-----	-----	------	--------

		346	
--	--	-----	--

LOC	OBJ	LINE	SOURCE
		347	\$EJECT
		348	;
		349	;*<
		350	; NAME:
		351	;
		352	; DESCRIPTION:
		353	; CALL:
		354	; ARGUMENTS:
		355	; MODIFIES:
		356	; RETURNS:
		357	; HISTORY:
		358	;*>
		359	;
002A		360	CMD_LOCK_PHONE:
002A		361	CMD_UNLOCK_PHONE:
002A		362	CMD_RESET_CALLS_POINTER:
002A		363	CMD_RESET_CALLS_COUNTER:
002A		364	CMD_RESET_AIR_TIME_METER:
		365	;
002A D200	F	366	setb CTI_HOST_CTS ; Flag host not ready for action!!!
		367	;
002C 120000	F	368	call CHECK_PHONE_STATUS
002F 300008	F	369	jnb LIVE_PHONE_IN_CRADDLE, CMD_PTR_STATUS_EXIT
0032 300005	F	370	jnb HOST_TURN_PHONE_ON, CMD_PTR_STATUS_EXIT
		371	;
0035 E500	F	372	mov a, HOST_CMD_REG
0037 120000	F	373	call TRANSFER_PHONE_DATA
		374	;
003A		375	CMD_PTR_STATUS_EXIT:
003A 22		376	ret
		377	

LOC	OBJ	LINE	SOURCE
		378	\$EJECT
		379	;
		380	;*<
		381	; NAME:
		382	;
		383	; DESCRIPTION:
		384	; CALL:
		385	; ARGUMENTS:
		386	; MODIFIES:
		387	; RETURNS:
		388	; HISTORY:
		389	;*>
		390	;
003B		391	CMD_WRITE_PHONE_TIME:
		392	;
003B	120000	F 393	call RECEIVE_HOST_DATA ; Receive PTR host data
		394	;
003E	120000	F 395	call CHECK_PHONE_STATUS
0041	300013	F 396	jnb LIVE_PHONE_IN_CRADDLE,
			CMD_WRITE_PHONE_TIME_EXIT
0044	300010	F 397	jnb HOST_TURN_PHONE_ON,
			CMD_WRITE_PHONE_TIME_EXIT
		398	;
0047	7809	399	mov r0, #09H ; Command + data byte count
0049	7900	F 400	mov r1, #low XFER_DATA_BUFFER ; Xfer buffer pointer
004B	E500	F 401	mov a, HOST_CMD_REG
		402	;
004D		403	CMD_WRITE_PHONE_TIME00:
004D	120000	F 404	call TRANSFER_PHONE_DATA
0050	200004	F 405	jb PTR_XFER_TIMEDOUT,
			CMD_WRITE_PHONE_TIME_EXIT
0053	E7	406	mov a, @r1
0054	09	407	inc r1
0055	D8F6	408	djnz r0, CMD_WRITE_PHONE_TIME00
		409	;
0057		410	CMD_WRITE_PHONE_TIME_EXIT:
0057	22	411	ret
		412	

LOC OBJ LINE SOURCE

```

413          $EJECT
414          ;
415          ;*<
416          ; NAME:
417          ;
418          ; DESCRIPTION:
419          ; CALL:
420          ; ARGUMENTS:
421          ; MODIFIES:
422          ; RETURNS:
423          ; HISTORY:
424          ;*>
425          ;
0058          426  CMD_READ_PHONE_CALLS:
427          ;
0058 D200    F 428          setb  HOST_XFER_ENABLED
005A 120000 F 429          call  SETUP_HOST_TIMEOUT
005D D200    F 430          setb  HOST_CMD_PARAM_XFER          ; Flag wait for params
005F D2A9          431          setb  XFER_TIMEOUT_INTERRUPT
432          ;
0061          433  CMD_READ_CALLS00:
0061 20006B F 434          jb     HOST_XFER_TIMEDOUT, CMD_READ_CALLS_EXIT
0064 2000FA F 435          jb     HOST_CMD_PARAM_XFER, CMD_READ_CALLS00
436          ;
0067 C2A9          437          clr   XFER_TIMEOUT_INTERRUPT
0069 C200    F 438          clr   HOST_CMD_PARAM_XFER          ; Got those params
006B C200    F 439          clr   HOST_XFER_TIMEDOUT          ; Clear host timed out
flag
006D C200    F 440          clr   HOST_XFER_ENABLED
006F C28C          441          clr   START_CTI_TIMEOUT
442          ;
0071 120000 F 443          call  CHECK_PHONE_STATUS          ; If the phone awake ???
0074 300058 F 444          jnb    LIVE_PHONE_IN_CRADDLE,
CMD_READ_CALLS_EXIT
0077 300055 F 445          jnb    HOST_TURN_PHONE_ON, CMD_READ_CALLS_EXIT
446          ;
007A 850000 F 447          mov    HOST_XFER_COUNT, HOST_CMD_DATA_CNT
007D 850000 F 448          mov    HOST_XFER_COUNT + 1, HOST_CMD_DATA_CNT +
1
0080 E500    F 449          mov    a, HOST_XFER_COUNT          ; Get LSB
0082 4500    F 450          orl    a, HOST_XFER_COUNT + 1      ; Or MSB
0084 6049          451          jz    CMD_READ_CALLS_EXIT      ; If zero xfer complete
452          ;
0086 1500    F 453          dec    HOST_XFER_COUNT          ; Subtract the so called
LRC
0088 C200    F 454          clr    HOST_XFER_COMPLETE
008A 750000 F 455          mov    PTR_CHECKSUM_REG, #00H      ; Clear CTPHONE
data checksum register
008D 750001 F 456          mov    CTI_BUFFER_COUNT, #01H      ; For multi-block xfers
457          ;
458          ; Issue the CTPHONE read command
459          ;
0090          460  CMD_READ_CALLS11:
0090 E500    F 461          mov    a, HOST_CMD_REG

```


0092 120000 F 462
command

call TRANSFER_PHONE_DATA ; Xfer the CTPHONE

0095 200037 F 463

jb PTR_XFER_TIMEOUT, CMD_READ_CALLS_EXIT

LOC OBJ LINE SOURCE

```

    464 ;
    465 ;   Read the CTPHONE returned data
    466 ;
0098    467  CMD_READ_CALLS22:
0098 120000 F 468          call RECEIVE_PHONE_DATA          ; A byte at a time
009B 200031 F 469          jb  PTR_XFER_TIMEDOUT, CMD_READ_CALLS_EXIT
    470 ;
009E 6200 F 471          xrl  PTR_CHECKSUM_REG, a          ; Compute the checksum,
as of old design
    472 ;
    473 ;   Transfer the CTPHONE returned data to the host
    474 ;
00A0 C298    475          clr  HOST_RXD                  ; Clear receive flag
00A2 C299    476          clr  HOST_TXD                  ; Clear transmit flag
00A4 F599    477          mov  HOST_DATA, a              ; Write that phone data byte to
the host
    478 ;
00A6 120000 F 479          call DEC_HOST_XFER_COUNT          ; Account for the
xfered byte
00A9 3000EC F 480          jnb  HOST_XFER_COMPLETE, CMD_READ_CALLS22
    481 ;
    482 ;   Time out for last byte xfer before so called LRC
    483 ;
00AC D200 F 484          setb  PTR_XFER_ENABLED          ; Use the phone timer
setup
00AE 7D33    485          mov  r5, #033H
00B0 7DF5    486          mov  r5, #0F5H
00B2 120000 F 487          call SETUP_TPHONE_TIMEOUT
00B5 D2A9    488          setb  XFER_TIMEOUT_INTERRUPT
    489 ;
00B7    490  CMD_READ_CALLS33:
00B7 3000FD F 491          jnb  PTR_XFER_TIMEDOUT, CMD_READ_CALLS33
00BA C2A9    492          clr  XFER_TIMEOUT_INTERRUPT
00BC C200 F 493          clr  PTR_XFER_TIMEDOUT
00BE C200 F 494          clr  PTR_XFER_ENABLED
    495 ;
    496 ;   Transfer the so called LRC?????
    497 ;
00C0    498  CMD_READ_CALLS44:
00C0 850000 F 499          mov  XFER_DATA_BUFFER, PTR_CHECKSUM_REG
00C3 750001 F 500          mov  CTI_BUFFER_COUNT, #01H
00C6 750001 F 501          mov  HOST_XFER_COUNT, #01H      ; Set LSB
00C9 750000 F 502          MOV  HOST_DATA_PTR, #low XFER_DATA_BUFFER
    503 ;
00CC 120000 F 504          call TRANSFER_HOST_DATA          ; Send the LRC???
    505 ;
00CF    506  CMD_READ_CALLS_EXIT:
00CF D200 F 507          setb  CTI_HOST_CTS              ; Flag host not ready for
action!!!
00D1 C200 F 508          clr  HOST_XFER_ENABLED
00D3 C2A9    509          clr  XFER_TIMEOUT_INTERRUPT
00D5 C28C    510          clr  START_CTI_TIMEOUT
00D7 22      511          ret
    512

```


LOC OBJ LINE SOURCE

```

513          $EJECT
514          ;
515          ;*<
516          ; NAME:
517          ;
518          ; DESCRIPTION:
519          ; CALL:
520          ; ARGUMENTS:
521          ; MODIFIES:
522          ; RETURNS:
523          ; HISTORY:
524          ;*>
525          ;
00D8          526  CMD_WRITE_TELEMAC_FIRMWARE:
527          ;
00D8 120000 F 528          call  SETUP_HOST_TIMEOUT
00DB D200 F 529          setb  HOST_CMD_PARAM_XFER          ; Flag wait for params
00DD D2A9          530          setb  XFER_TIMEOUT_INTERRUPT
531          ;
00DF          532  CMD_WRITE_FIRMWARE00:
00DF 200042 F 533          jb  HOST_XFER_TIMEDOUT,
CMD_WRITE_FIRMWARE_EXIT
00E2 2000FA F 534          jb  HOST_CMD_PARAM_XFER,
CMD_WRITE_FIRMWARE00
535          ;
00E5 C2A9          536          clr  XFER_TIMEOUT_INTERRUPT
00E7 C200 F 537          clr  HOST_CMD_PARAM_XFER          ; Got those params
00E9 C200 F 538          clr  HOST_XFER_TIMEDOUT          ; Clear host timed out
flag
00EB C200 F 539          clr  HOST_XFER_ENABLED
00ED D200 F 540          setb  CTI_HOST_CTS          ; Flag host not ready for
action!!!
541          ;
00EF 120000 F 542          call  CHECK_PHONE_STATUS          ; If the phone awake ???
00F2 30002F F 543          jnb  LIVE_PHONE_IN_CRADDLE,
CMD_WRITE_FIRMWARE_EXIT
00F5 30002C F 544          jnb  HOST_TURN_PHONE_ON,
CMD_WRITE_FIRMWARE_EXIT
545          ;
00F8 C200 F 546          clr  HOST_XFER_COMPLETE          ; Flag beginning xfers
00FA 850000 F 547          mov  HOST_XFER_COUNT, HOST_CMD_DATA_CNT
00FD 850000 F 548          mov  HOST_XFER_COUNT + 1, HOST_CMD_DATA_CNT +
1
0100 E500 F 549          mov  a, HOST_XFER_COUNT          ; Get LSB
0102 4500 F 550          orl  a, HOST_XFER_COUNT + 1          ; Or MSB
0104 601E          551          jz  CMD_WRITE_FIRMWARE_EXIT          ; If zero xfer
complete
552          ;
553          ; Issue the CTPHONE write firmware command
554          ;
0106 E500 F 555          mov  a, HOST_CMD_REG
0108 120000 F 556          call  TRANSFER_PHONE_DATA          ; Xfer the CTPHONE
command
```


010B 200016 F 557 jb PTR_XFER_TIMEDOUT,
CMD_WRITE_FIRMWARE_EXIT

558 ;

010E 559 CMD_WRITE_FIRMWARE11:

560 ;

561 ; Write those CTPHONE firmware bytes

562 ;

010E 563 CMD_WRITE_FIRMWARE22:

LOC	OBJ	LINE	SOURCE
010E	750001	F 564	mov CTI_BUFFER_COUNT, #01H ; Single byte for long xfers
0111	120000	F 565	call RECEIVE_HOST_DATA ; Get another firmware byte
0114	20000D	F 566	jb HOST_XFER_TIMEDOUT, CMD_WRITE_FIRMWARE_EXIT
		567	;
0117	E500	F 568	mov a, XFER_DATA_BUFFER
0119	120000	F 569	call TRANSFER_PHONE_DATA ; Xfer the CTPHONE data
011C	200005	F 570	jb PTR_XFER_TIMEDOUT, CMD_WRITE_FIRMWARE_EXIT
		571	;
011F	200002	F 572	jb HOST_XFER_COMPLETE, CMD_WRITE_FIRMWARE_EXIT
0122	80EA	573	jmp CMD_WRITE_FIRMWARE11
		574	;
0124		575	CMD_WRITE_FIRMWARE_EXIT:
0124	22	576	ret
		577	

LOC	OBJ	LINE	SOURCE
		578	\$EJECT
		579	;
		580	;*<
		581	; NAME:
		582	;
		583	; DESCRIPTION:
		584	; CALL:
		585	; ARGUMENTS:
		586	; MODIFIES:
		587	; RETURNS:
		588	; HISTORY:
		589	;*>
		590	;
0125		591	UPDATE_HOST_XFER_STATUS:
		592	;
0125	750001	F 593	mov CTI_BUFFER_COUNT, #01H ; For multi-block xfers
0128	E500	F 594	mov a, HOST_XFER_COUNT ; Get LSB
012A	4500	F 595	orl a, HOST_XFER_COUNT + 1 ; Or MSB
012C	7002	596	jnz UPDATE_HOST_XFER_STATUS_EXIT ; If zero xfer complete
		597	;
012E	D200	F 598	setb HOST_XFER_COMPLETE
		599	;
0130		600	UPDATE_HOST_XFER_STATUS_EXIT:
0130	22	601	ret
		602	

LOC	OBJ	LINE	SOURCE
		603	\$EJECT
		604	;
		605	;*<
		606	; NAME:
		607	;
		608	; DESCRIPTION:
		609	; CALL:
		610	; ARGUMENTS:
		611	; MODIFIES:
		612	; RETURNS:
		613	; HISTORY:
		614	;*>
		615	;
0131		616	CMD_PHONE_IN_CRADDLE:
		617	;
0131	120000	F 618	call CHECK_PHONE_STATUS
		619	;
0134	A200	F 620	mov c, LIVE_PHONE_IN_CRADDLE
0136	9200	F 621	mov CTI_ACTIVE_CTPHONE, c
		622	;
0138	850000	F 623	mov HOST_XFER_COUNT, CTI_BUFFER_COUNT
013B	750000	F 624	mov HOST_XFER_COUNT + 1, #00H
013E	850000	F 625	mov XFER_DATA_BUFFER, HOST_CTI_STATUS
		626	;
0141	120000	F 627	call TRANSFER_HOST_DATA
		628	;
0144	22	629	ret
		630	

LOC OBJ LINE SOURCE

```

        631          $EJECT
        632      ;
        633      ;*<
        634      ;   NAME:
        635      ;
        636      ;   DESCRIPTION:
        637      ;   CALL:
        638      ;   ARGUMENTS:
        639      ;   MODIFIES:
        640      ;   RETURNS:
        641      ;   HISTORY:
        642      ;*>
        643      ;
0145      644  CMD_FAKE_POWER_DOWN:
        645      ;
0145 D200  F  646          setb  CTI_HOST_CTS          ; Flag host not ready for
action!!!
0147 C2A8      647          clr   PTR_800_INTERRUPT      ; Turn off the phone interrupt
        648      ;
0149 E500  F  649          mov   a, HOST_CMD_REG
014B 120000  F  650          call  TRANSFER_PHONE_DATA
014E 200003  F  651          jb    PTR_XFER_TIMEDOUT,
CMD_FAKE_POWER_DOWN_EXIT
        652      ;
0151 120000  F  653          call  DELAY_350_MSECS      ; Wait for 231 msecs
        654      ;
0154      655  CMD_FAKE_POWER_DOWN_EXIT:
0154 020000  F  656          ljmp  CMD_TURN_PHONE_OFF
        657      ;
0157 22      658          ret
        659

```


LOC OBJ LINE SOURCE

```

        660          $EJECT
        661      ;
        662      ;*<
        663      ; NAME:
        664      ;
        665      ; DESCRIPTION:
        666      ; CALL:
        667      ; ARGUMENTS:
        668      ; MODIFIES:
        669      ; RETURNS:
        670      ; HISTORY:
        671      ;*>
        672      ;
0158      673  CMD_TURN_PHONE_OFF:
        674      ;
0158 D200  F  675          setb  CTI_HOST_CTS          ; Flag host not ready for
action!!!
015A C200  F  676          clr   CTI_PTR_EXPWR_ON      ; Turn of the phone
015C C200  F  677          clr   HOST_TURN_PHONE_ON
015E C200  F  678          clr   CTI_TURN_OFF_PHONE
0160 C200  F  679          clr   CTI_LED_1
        680      ;
0162 120000 F  681          call  DELAY_350_MSECS      ; Wait for ~350 msecs
0165 120000 F  682          call  DELAY_350_MSECS      ; Wait for ~350 msecs
0168 120000 F  683          call  DELAY_350_MSECS      ; Wait for ~350 msecs
        684      ;
016B 22      685          ret
        686
```


LOC	OBJ	LINE	SOURCE
		687	\$EJECT
		688	;
		689	;*<
		690	NAME:
		691	;
		692	DESCRIPTION:
		693	CALL:
		694	ARGUMENTS:
		695	MODIFIES:
		696	RETURNS:
		697	HISTORY:
		698	;*>
		699	;
016C		700	CMD_TURN_POWER_ON:
		701	;
016C D200	F	702	setb CTI_HOST_CTS ; Flag host not ready for action!!!
016E D200	F	703	setb HOST_TURN_PHONE_ON
0170 C200	F	704	clr CTI_TURN_OFF_PHONE
0172 D200	F	705	setb CTI_PTR_EXPWR_ON ; Turn on the phone
0174 D200	F	706	setb CTI_LED_1
		707	;
0176 120000	F	708	call DELAY_350_MSECS ; Wait for ~350 msecs
0179 120000	F	709	call DELAY_350_MSECS ; Wait for ~350 msecs
017C 120000	F	710	call DELAY_350_MSECS ; Wait for ~350 msecs
		711	;
017F		712	CMD_TURN_POWER_ON_EXIT:
017F 22		713	ret
		714	

LOC	OBJ	LINE	SOURCE
		715	\$EJECT
		716	;
		717	;*<
		718	; NAME:
		719	;
		720	; DESCRIPTION:
		721	; CALL:
		722	; ARGUMENTS:
		723	; MODIFIES:
		724	; RETURNS:
		725	; HISTORY:
		726	;*>
		727	;
0180		728	CMD_READ_CTI_VERSION:
		729	;
0180 D200	F	730	setb CTI_HOST_CTS ; Flag host not ready for action!!!
0182 758300	F	731	mov dph, #high CTI_FIRMWARE_REVISION
0185 758200	F	732	mov dpl, #low CTI_FIRMWARE_REVISION
0188 850000	F	733	mov HOST_XFER_COUNT, CTI_BUFFER_COUNT
018B 750000	F	734	mov HOST_XFER_COUNT + 1, #00H
		735	;
018E A800	F	736	mov r0, CTI_BUFFER_COUNT
0190 7900	F	737	mov r1, #low XFER_DATA_BUFFER ; Xfer buffer pointer
0192 7A00		738	mov r2, #00H
		739	;
0194		740	CMD_READ_CTI_VERSION00:
		741	;
0194 EA		742	mov a, r2 ; Current xfer byte
0195 93		743	movc a, @a+dprr
0196 F7		744	mov @r1, a
0197 0A		745	inc r2
0198 09		746	inc r1
0199 D8F9		747	djnz r0, CMD_READ_CTI_VERSION00
		748	;
019B 120000	F	749	call TRANSFER_HOST_DATA
		750	;
019E 22		751	ret
		752	

400

LOC OBJ LINE SOURCE

```

    753          $EJECT
    754      ;
    755      ;*<
    756      ; NAME:
    757      ;
    758      ; DESCRIPTION:
    759      ; CALL:
    760      ; ARGUMENTS:
    761      ; MODIFIES:
    762      ; RETURNS:
    763      ; HISTORY:
    764      ;*>
    765      ;
    766      ;
019F    767 CTI_FIRMWARE_REVISION:
    768      ;
019F 43544920 769          DB 'CTI VER.:1.0 AUG-20-1991'
01A3 5645522E
01A7 3A312E30
01AB 20204155
01AF 472D3230
01B3 2D313939
01B7 31
    770
    771          END
    ;
    ; End of CTI_CMDS.A51
    ;
```

REGISTER BANK(S) USED: 0

ASSEMBLY COMPLETE, NO ERRORS FOUND

MS-DOS MACRO ASSEMBLER A51 V4.4
OBJECT MODULE PLACED IN CTI_CNST.OBJ
ASSEMBLER INVOKED BY: A51 CTI_CNST.INC

LOC	OBJ	LINE	SOURCE
		1 ;	\$PAGEWIDTH (127)
		2 ;	\$PAGELENGTH (57)
		3 ;	
		4 ;	\$TITLE (CTI_CNST.INC)
		5 ;	
		6 ;	Program Title: Cellular Telephone Interface Controller Firmwa
		7 ;	Filename : CTI_CNST.INC
		8 ;	Project # :
		9 ;	Author : Theodore W. Watler
		10 ;	From : Parchment Designs
		11 ;	For : Turner, Gold, France & Associates
		12 ;	Date Created : August 2, 1991
		13 ;	Version : A.00
		14 ;	
		15 ;	
		16 ;	
		17 ;	COPYRIGHT (C) 1991. ALL RIGHTS RESERVED
		18 ;	Turner, Gold, France & Associates
		19 ;	
		20 ;	
		21 ;	
		22 ;	PROGRAM FUNCTION
		23 ;	
		24 ;	
		25 ;	PROGRAM DESCRIPTION
		26 ;	
		27 ;	
		28 ;	REFERENCES
		29 ;	
		30 ;	1. 8051 Hardware Reference Manual
		31 ;	2. Franklin Software DK51 Development Tools
		32 ;	3.
		33 ;	
		34 ;	***** MODULE HISTORY *****
		35 ;	
		36	
			#####
		37	

LOC OBJ	LINE	SOURCE
	38	\$EJECT
	39	;
	40	; Register Banks Identification Constants
	41	;
0000	42	REG_BANK_00 EQU 00H ; BANK 0 SELECT
0001	43	REG_BANK_01 EQU 01H ; BANK 1 SELECT
0002	44	REG_BANK_02 EQU 02H ; BANK 2 SELECT
0003	45	REG_BANK_03 EQU 03H ; BANK 3 SELECT
	46	;
0000	47	REGISTER_00 EQU 00H ; R0 direct address
	48	;
	49	; CTI Internal Test Error Equates
	50	;
0000	51	M8031_FAULT EQU 00H
0001	52	PROM_FAULT EQU 01H
	53	;
	54	; INTERRUPT ENABLE/DISABLE EQUATES (INTERRUPT ENABLE REGISTER, IE)
	55	;
00A8	56	PTR_800_INTERRUPT EQU EX0
00A9	57	XFER_TIMEOUT_INTERRUPT EQU ET0
00AC	58	HOST_COMM_INTERRUPT EQU ES
00AF	59	ENABLE_ALL_INTERRUPT EQU EA
	60	;
	61	; INTERRUPT EQUATES (TIMER/COUNTER CONTROL REGISTER, TCON)
	62	;
0098	63	HOST_RXD EQU RI
0099	64	HOST_TXD EQU TI
0099	65	HOST_DATA EQU SBUF
F4C0 BYTE	66	HOST_TIMEOUT_CNT EQU 0F4C0H ; 3.125 msecs per
	67	;
DC00	68	PTR_TIMEOUT_CNT EQU 0DC00H ; PTR xfer 10msecs TO
EE00	69	PTR_EXCLK_TIMEOUT_CNT EQU 0EE00H ; PTR 5msecs
EXCLK TO		
0023	70	PTR_EXCLK_DEBOUNCE EQU 023H ; Debounce pulse count
008C	71	START_CTI_TIMEOUT EQU TR0
	72	;
00FD	73	BAUD_9600 EQU 0FDH
00FA	74	BAUD_4800 EQU 0FAH
0038	75	MAX_BUFFER_SIZE EQU 038H
	76	

LOC OBJ LINE SOURCE

```

77          $EJECT
78 ;
79 ; Host to CTI Command Equates
80 ;
0000      81 CTI_RD_PHONE_NUMBER EQU 000H
0001      82 CTI_RD_PHONE_CALLS EQU 001H
0002      83 CTI_RD_PHONE_TIME EQU 002H
0003      84 CTI_WR_PHONE_TIME EQU 003H
0004      85 CTI_TBD_04 EQU 004H
0005      86 CTI_TURN_PHONE_OFF EQU 005H
0006      87 CTI_RD_PHONE_RTb_VER EQU 006H
0007      88 CTI_RD_NOVATEL_VER EQU 007H
0008      89 CTI_RD_MBC_VERSION EQU 008H
0009      90 CTI_TURN_POWER_ON EQU 009H
000A      91 CTI_LOCK_PHONE EQU 00AH
000B      92 CTI_UNLOCK_PHONE EQU 00BH
000C      93 CTI_TBD_12 EQU 00CH
000D      94 CTI_RD_AIR_TIME_METER EQU 00DH
000E      95 CTI_FAKE_POWER_DOWN EQU 00EH
000F      96 CTI_RD_CALLS_COUNTER EQU 00FH
0010      97 CTI_RD_CALLS_RAM_PTR EQU 010H
0011      98 CTI_WR_TELEMAC_FIRMWARE EQU 011H
0012      99 CTI_PHONE_IN_CRADDLE EQU 012H
0013     100 CTI_RESET_CALLS_POINTER EQU 013H
0014     101 CTI_RESET_CALLS_COUNTER EQU 014H
0015     102 CTI_RESET_AIR_TIME_METER EQU 015H
0016     103 CTI_MAX_COMMAND_COUNT EQU CTI_RESET_AIR_TIME_METER +
1
104 ;
105 ; End of CTI_CNST.INC
106 ;
107

```


SYMBOL TABLE LISTING

NAME	TYPE	VALUE	ATTRIBUTES
BAUD_4800.....	N NUMB	00FAH	A
BAUD_9600.....	N NUMB	00FDH	A
CTI_FAKE_POWER_DOWN...	N NUMB	000EH	A
CTI_LOCK_PHONE.....	N NUMB	000AH	A
CTI_MAX_COMMAND_COUNT..	N NUMB	0016H	A
CTI_PHONE_IN_CRADDLE..	N NUMB	0012H	A
CTI_RD_AIR_TIME_METER..	N NUMB	000DH	A
CTI_RD_CALLS_COUNTER..	N NUMB	000FH	A
CTI_RD_CALLS_RAM_PTR..	N NUMB	0010H	A
CTI_RD_MBC_VERSION...	N NUMB	0008H	A
CTI_RD_NOVATEL_VER...	N NUMB	0007H	A
CTI_RD_PHONE_CALLS...	N NUMB	0001H	A
CTI_RD_PHONE_NUMBER...	N NUMB	0000H	A
CTI_RD_PHONE_RTb_VER..	N NUMB	0006H	A
CTI_RD_PHONE_TIME....	N NUMB	0002H	A
CTI_RESET_AIR_TIME_METER	N NUMB	0015H	A
CTI_RESET_CALLS_COUNTER.	N NUMB	0014H	A
CTI_RESET_CALLS_POINTER.	N NUMB	0013H	A
CTI_TBD_04.....	N NUMB	0004H	A
CTI_TBD_12.....	N NUMB	000CH	A
CTI_TURN_PHONE_OFF...	N NUMB	0005H	A
CTI_TURN_POWER_ON....	N NUMB	0009H	A
CTI_UNLOCK_PHONE.....	N NUMB	000BH	A
CTI_WR_PHONE_TIME....	N NUMB	0003H	A
CTI_WR_TELEMAC_FIRMWARE.	N NUMB	0011H	A
EA.....	B ADDR	00A8H.7	A
ENABLE_ALL_INTERRUPT..	B ADDR	00A8H.7	A
ES.....	B ADDR	00A8H.4	A
ET0.....	B ADDR	00A8H.1	A
EX0.....	B ADDR	00A8H.0	A
HOST_COMM_INTERRUPT...	B ADDR	00A8H.4	A
HOST_DATA.....	D ADDR	0099H	A
HOST_RXD.....	B ADDR	0098H.0	A
HOST_TIMEOUT_CNT....	N NUMB	F4C0H	A
HOST_TXD.....	B ADDR	0098H.1	A
M8031_FAULT.....	N NUMB	0000H	A
MAX_BUFFER_SIZE.....	N NUMB	0038H	A
PROM_FAULT.....	N NUMB	0001H	A
PTR_800_INTERRUPT....	B ADDR	00A8H.0	A
PTR_EXCLK_DEBOUNCE...	N NUMB	0023H	A
PTR_EXCLK_TIMEOUT_CNT..	N NUMB	EE00H	A
PTR_TIMEOUT_CNT.....	N NUMB	DC00H	A
REGISTER_00.....	N NUMB	0000H	A
REG_BANK_00.....	N NUMB	0000H	A
REG_BANK_01.....	N NUMB	0001H	A
REG_BANK_02.....	N NUMB	0002H	A
REG_BANK_03.....	N NUMB	0003H	A
RI.....	B ADDR	0098H.0	A
SBUF.....	D ADDR	0099H	A
START_CTI_TIMEOUT....	B ADDR	0088H.4	A

TI..... B ADDR 0098H.1 A
TR0..... B ADDR 0088H.4 A
XFER_TIMEOUT_INTERRUPT. B ADDR 00A8H.1 A

REGISTER BANK(S) USED: 0

ASSEMBLY COMPLETE, NO ERRORS FOUND

A51 MACRO ASSEMBLER CTI_GLBL.A51
27/09/91 PAGE 1

DATE

MS-DOS MACRO ASSEMBLER A51 V4.4
OBJECT MODULE PLACED IN CTI_GLBL.OBJ
ASSEMBLER INVOKED BY: A51 CTI_GLBL.A51 DEBUG ERRORPRINT(CTI_GLBL.ERR)
NOSYMBOLS NOXREF

LOC	OBJ	LINE	SOURCE
		1	\$PAGEWIDTH (127)
		2	\$PAGELENGTH (57)
		3	;
		4	\$TITLE (CTI_GLBL.A51)
		5	;
		6	;
		7	Program Title: Cellular Telephone Interface Controller Firmwa
		8	Filename : CTI_GLBL.A51
		9	Module Name : CTI_GLBL.OBJ
		10	Project # :
		11	Author : Theodore W. Watler
		12	From : Parchment Designs
		13	For : Turner, Gold, France & Associates
		14	Date Created : August 3, 1991
		15	Version : A.00
		16	;
		17	;
		18	COPYRIGHT (C) 1991. ALL RIGHTS RESERVED
		19	Turner, Gold, France & Associates
		20	;
		21	;
		22	;
		23	PROGRAM FUNCTION
		24	;
		25	;
		26	PROGRAM DESCRIPTION
		27	;
		28	;
		29	REFERENCES
		30	;
		31	1. 8051 Hardware Reference Manual
		32	2. Franklin Software DK51 Development Tools
		33	3.
		34	;
		35	***** MODULE HISTORY *****
		36	;
		37	;
		38	#####

LOC	OBJ	LINE	SOURCE
		39	\$EJECT
		40	;
		41	NAME CTI_GLOBAL_VARIABLES
		42	;
		43	;
		44	;
		45	PUBLIC CTI_HOST_RTS
		46	PUBLIC CTI_HOST_CTS
		47	PUBLIC CTI_PTR_RXD
		48	PUBLIC CTI_PTR_EXPWR_ON
		49	PUBLIC CTI_PTR_TXD
		50	PUBLIC CTI_LED_1
		51	
		52	PUBLIC CTI_LED_2
		53	PUBLIC CTI_PTR_EXCLK
		54	PUBLIC CTI_HOST_TXD
		55	PUBLIC CTI_HOST_RXD
		56	
		57	PUBLIC DELAY_CTR_02
		58	PUBLIC DELAY_CTR_01
		59	PUBLIC DELAY_CTR_00
		60	
		61	PUBLIC HOST_CMD_REG
		62	PUBLIC HOST_CMD_DATA_CNT
		63	PUBLIC HOST_DATA_PTR
		64	PUBLIC HOST_XFER_COUNT
		65	PUBLIC CTI_BUFFER_COUNT
		66	PUBLIC PTR_BIT_COUNT
		67	PUBLIC PTR_PULSE_COUNT
		68	PUBLIC PTR_CHECKSUM_REG
		69	
		70	PUBLIC CTI_STATUS
		71	PUBLIC CTI_TURN_OFF_PHONE
		72	PUBLIC LIVE_PHONE_IN_CRADDLE
		73	PUBLIC HOST_TURN_PHONE_ON
		74	PUBLIC XFER_BUFFER_FULL
		75	PUBLIC HOST_COMMAND
		76	
		77	PUBLIC HOST_CTI_STATUS
		78	PUBLIC CTI_CTPHONE_TIMEDOUT
		79	PUBLIC CTI_HOST_TIMEDOUT
		80	PUBLIC CTI_ACTIVE_CTPHONE
		81	
		82	PUBLIC HOST_XFER_STATUS
		83	PUBLIC HOST_CMD_PARAM_XFER
		84	PUBLIC HOST_XFER_COMPLETE
		85	PUBLIC HOST_XFER_TIMEDOUT
		86	PUBLIC HOST_XFER_ENABLED
		87	
		88	PUBLIC PTR_XFER_STATUS
		89	PUBLIC PTR_EDGE_DETECTED

LOC	OBJ	LINE	SOURCE
		90	PUBLIC PTR_ONLINE_CHECK
		91	PUBLIC PTR_DATA_TRANSMITTED
		92	PUBLIC PTR_XFER_TIMEDOUT
		93	PUBLIC PTR_XFER_ENABLED
		94	
		95	PUBLIC XFER_DATA_BUFFER
		96	;
		206	\$LIST
		207	

LOC OBJ LINE SOURCE

```

208          $EJECT
209 ;
210 ; 8031 PORT 0 I/O DEFINITIONS
211 ;
212          ; ADDRESS/DATA BUS ----- BIT(7)
213          ; ADDRESS/DATA BUS ----- BIT(6)
214          ; ADDRESS/DATA BUS ----- BIT(5)
215          ; ADDRESS/DATA BUS ----- BIT(4)
216          ; ADDRESS/DATA BUS ----- BIT(3)
217          ; ADDRESS/DATA BUS ----- BIT(2)
218          ; ADDRESS/DATA BUS ----- BIT(1)
219          ; ADDRESS/DATA BUS ----- BIT(0)
220 ;
221 ; 8031 PORT 1 I/O DEFINITIONS
222 ;
223          ; Unused ----- Bit(7)
224          ; Unused ----- Bit(6)
0095      225 CTI_HOST_RTS      BIT P1.5 ;
0094      226 CTI_HOST_CTS      BIT P1.4 ;
0093      227 CTI_PTR_RXD       BIT P1.3 ;
0092      228 CTI_PTR_EXPWR_ON   BIT P1.2 ;
0091      229 CTI_PTR_TXD       BIT P1.1 ;
0090      230 CTI_LED_1         BIT P1.0 ;
231 ;
232 ; 8031 PORT 2 I/O DEFINITIONS
233 ;
234          ; ADDRESS BUS ----- BIT(15)
235          ; ADDRESS BUS ----- BIT(14)
236          ; ADDRESS BUS ----- BIT(13)
237          ; ADDRESS BUS ----- BIT(12)
238          ; ADDRESS BUS ----- BIT(11)
239          ; ADDRESS BUS ----- BIT(10)
240          ; ADDRESS BUS ----- BIT(9)
241          ; ADDRESS BUS ----- BIT(8)
242 ;
243 ; 8031 PORT 3 I/O DEFINITIONS
244 ;
00B7      245 CTI_LED_2         BIT P3.7 ; CTI LED 2 ----- Bit(7)
246          ; Unused ----- Bit(6)
247          ; Unused ----- Bit(5)
248          ; Unused ----- Bit(4)
249          ; Unused ----- Bit(3)
00B2      250 CTI_PTR_EXCLK     BIT P3.2 ; PTR-800 external clk input
(500HZ) Bit(2)
00B1      251 CTI_HOST_TXD     BIT P3.1 ; Host Serial Transmit Int -----
Bit(1)
00B0      252 CTI_HOST_RXD     BIT P3.0 ; Host Serial Receive Int -----
Bit(0)
253

```


LOC OBJ LINE SOURCE

```
254          $EJECT
255 ;
256 ;
257 ; REGISTER BANK 0
258 ;
259          ; General Purpose Register 00
260          ; General Purpose Register 01
261          ; General Purpose Register 02
262          ; General Purpose Register 03
263          ; General Purpose Register 04
264          ; General Purpose Register 05
265          ; General Purpose Register 06
266          ; General Purpose Register 07
267 ;
268 ; REGISTER BANK 1
269 ;
270          ; General Purpose Register 10
271          ; General Purpose Register 11
272          ; General Purpose Register 12
273          ; General Purpose Register 13
274          ; General Purpose Register 14
275          ; General Purpose Register 15
276          ; General Purpose Register 16
277          ; General Purpose Register 17
278 ;
279 ; REGISTER BANK 2
280 ;
281          ; General Purpose Register 20
282          ; General Purpose Register 21
283          ; General Purpose Register 22
284          ; General Purpose Register 23
285          ; Assigned to data space!!!!
286          ; Assigned to data space!!!!
287          ; Assigned to data space!!!!
288          ; Assigned to data space!!!!
289
```


LOC OBJ LINE SOURCE

```

290          $EJECT
291 ;
292          DSEG AT 13H
293 ;
0013      294  DELAY_CTR_02:      DS  1
0014      295  DELAY_CTR_01:      DS  1
0015      296  DELAY_CTR_00:      DS  1
297
0016      298  HOST_CMD_REG:      DS  1
0017      299  HOST_CMD_DATA_CNT:  DS  2
0019      300  HOST_DATA_PTR:      DS  1
001A      301  HOST_XFER_COUNT:    DS  2
302
001C      303  CTI_BUFFER_COUNT:   DS  1
001D      304  PTR_BIT_COUNT:      DS  1
001E      305  PTR_PULSE_COUNT:    DS  1
001F      306  PTR_CHECKSUM_REG:   DS  1
307 ;
308          DSEG AT 20H
309 ;
310 ;    BIT ADDRESSABLE SEGMENT
311 ;
0020      312  CTI_STATUS:         DS  1
0004      313  CTI_TURN_OFF_PHONE BIT  CTI_STATUS.4 ;
0003      314  LIVE_PHONE_IN_CRADDLE BIT  CTI_STATUS.3 ;
0002      315  HOST_TURN_PHONE_ON  BIT  CTI_STATUS.2 ;
0001      316  XFER_BUFFER_FULL    BIT  CTI_STATUS.1 ;
0000      317  HOST_COMMAND        BIT  CTI_STATUS.0 ;
318 ;
0021      319  HOST_CTI_STATUS:    DS  1
000A      320  CTI_CTPHONE_TIMEDOUT BIT  HOST_CTI_STATUS.2
0009      321  CTI_HOST_TIMEDOUT   BIT  HOST_CTI_STATUS.1
0008      322  CTI_ACTIVE_CTPHONE  BIT  HOST_CTI_STATUS.0
323 ;
0022      324  HOST_XFER_STATUS:    DS  1
0013      325  HOST_CMD_PARAM_XFER BIT  HOST_XFER_STATUS.3
0012      326  HOST_XFER_COMPLETE  BIT  HOST_XFER_STATUS.2
0011      327  HOST_XFER_TIMEDOUT  BIT  HOST_XFER_STATUS.1
0010      328  HOST_XFER_ENABLED   BIT  HOST_XFER_STATUS.0
329 ;
0023      330  PTR_XFER_STATUS:     DS  1
001C      331  PTR_EDGE_DETECTED   BIT  PTR_XFER_STATUS.4
001B      332  PTR_ONLINE_CHECK    BIT  PTR_XFER_STATUS.3
001A      333  PTR_DATA_TRANSMITTED BIT  PTR_XFER_STATUS.2
0019      334  PTR_XFER_TIMEDOUT   BIT  PTR_XFER_STATUS.1
0018      335  PTR_XFER_ENABLED    BIT  PTR_XFER_STATUS.0
336 ;
337          DSEG AT 28H
338 ;
0028      339  XFER_DATA_BUFFER:    DS  38H      ; 56 bytes fifo to/from PTR-800
340

```


LOC	OBJ	LINE	SOURCE
		341	END
			;
			; End of CTI_GLBL.A51
			;

REGISTER BANK(S) USED: 0

ASSEMBLY COMPLETE, NO ERRORS FOUND

LOC	OBJ	LINE	SOURCE
		39	\$EJECT
		40	;
		41	NAME CTI_INTERRUPT_SERVICES
		42	;
		43	EXTERNAL REFERENCE TABLE
		44	;
		45	EXTRN CODE (DEC_HOST_XFER_COUNT)
		46	EXTRN CODE (SETUP_HOST_TIMEOUT)
		47	
		48	EXTRN IDATA(STACK)
		49	;
		50	PUBLIC DECLARATION TABLE
		51	;
		52	PUBLIC CTPHONE_ACTIVE_ISR
		53	PUBLIC CTI_TIMEOUT_ISR
		54	PUBLIC HOST_XFER_ISR
		55	;
		262	\$LIST
		263	;
		264	
		265	CTI_INTERRUPTS SEGMENT CODE
----		266	RSEG CTI_INTERRUPTS
		267	USING REG_BANK_00
		268	

LOC	OBJ	LINE	SOURCE
		269	\$EJECT
		270	;
		271	;*<
		272	NAME:
		273	;
		274	DESCRIPTION:
		275	CALL:
		276	ARGUMENTS:
		277	MODIFIES:
		278	RETURNS:
		279	HISTORY:
		280	;*>
		281	;
0000		282	CTPHONE_ACTIVE_ISR:
		283	;
0000	C0E0	284	push acc
0002	C0D0	285	push psw
		286	;
0004	D200 F	287	setb PTR_EDGE_DETECTED ; We have a pulse
0006	0500 F	288	inc PTR_PULSE_COUNT
0008	E500 F	289	mov a, PTR_PULSE_COUNT
000A	B4230D	290	cjne a, #PTR_EXCLK_DEBOUNCE,
			CTPHONE_ACTIVE_ISR_EXIT
000D	750000 F	291	mov PTR_PULSE_COUNT, #00H ; Clear the EXCLK
			pulse cnt
		292	;
0010	300005 F	293	jnb PTR_ONLINE_CHECK, CTPHONE_ACTIVE_ISR00
0013	D200 F	294	setb LIVE_PHONE_IN_CRADDLE
0015	020000 F	295	jmp CTPHONE_ACTIVE_ISR_EXIT
		296	;
0018		297	CTPHONE_ACTIVE_ISR00:
		298	jb HOST_TURN_PHONE_ON, CTPHONE_ACTIVE_ISR_EXIT
0018	D205	299	setb CTI_TURN_PHONE_OFF
		300	;
001A		301	CTPHONE_ACTIVE_ISR_EXIT:
001A	D0D0	302	pop psw
001C	D0E0	303	pop acc
001E	32	304	reti
		305	

LOC	OBJ	LINE	SOURCE
		306	\$EJECT
		307	;
		308	;*<
		309	; NAME:
		310	;
		311	; DESCRIPTION:
		312	; CALL:
		313	; ARGUMENTS:
		314	; MODIFIES:
		315	; RETURNS:
		316	; HISTORY:
		317	;*>
		318	;
001F		319	CTI_TIMEOUT_ISR:
		320	;
001F	200006 F	321	jb HOST_XFER_ENABLED, CTI_TIMEOUT_ISR00
0022	200008 F	322	jb PTR_XFER_ENABLED, CTI_TIMEOUT_ISR11
		323	;
0025	020000 F	324	jmp CTI_TIMEOUT_ISR_EXIT
		325	;
0028		326	CTI_TIMEOUT_ISR00:
0028	D200 F	327	setb HOST_XFER_TIMEDOUT
002A	020000 F	328	jmp CTI_TIMEOUT_ISR_EXIT
		329	;
002D		330	CTI_TIMEOUT_ISR11:
002D	D200 F	331	setb PTR_XFER_TIMEDOUT
		332	;
002F		333	CTI_TIMEOUT_ISR_EXIT:
002F	32	334	reti
		335	

417

LOC	OBJ	LINE	SOURCE
		336	\$EJECT
		337	;
		338	;* <
		339	; NAME:
		340	;
		341	; DESCRIPTION:
		342	; CALL:
		343	; ARGUMENTS:
		344	; MODIFIES:
		345	; RETURNS:
		346	; HISTORY:
		347	;* >
		348	;
0030		349	HOST_XFER_ISR:
		350	;
0030 C0E0		351	push acc ; Save the accumulator
0032 C0D0		352	push psf ; Save the status word
0034 C000		353	push REGISTER_00
		354	;
0036 109806		355	jbc HOST_RXD, RECEIVE_HOST_DATA_ISR
0039 109952		356	jbc HOST_TXD, TRANSMIT_HOST_DATA_ISR
003C 020000 F		357	jmp HOST_XFER_ISR_EXIT ; Exit this isr
		358	;
003F		359	RECEIVE_HOST_DATA_ISR:
		360	;
003F 200024 F		361	jb HOST_COMMAND, RECEIVE_HOST_DATA_ISR00
		362	;
0042 859900 F		363	mov HOST_CMD_REG, HOST_DATA ; Get the transmitted command
0045 D200 F		364	setb HOST_COMMAND ; First host xfer is command byte
0047 C200 F		365	clr HOST_XFER_TIMEOUT
0049 C200 F		366	clr HOST_CMD_PARAM_XFER
		367	;
004B 750000 F		368	mov HOST_DATA_PTR, #low XFER_DATA_BUFFER
004E 750000 F		369	mov HOST_CMD_DATA_CNT, #00H ; Number of bytes the host will xfer
0051 750000 F		370	mov HOST_CMD_DATA_CNT + 1, #00H
0054 750000 F		371	mov HOST_XFER_COUNT, #00H ; Clear the host byte count lsb
0057 750000 F		372	mov HOST_XFER_COUNT + 1, #00H ; Clear the host byte count msb
		373	;
005A 750000 F		374	mov CTI_BUFFER_COUNT, #00H ; Clear the current fifo count
005D C200 F		375	clr XFER_BUFFER_FULL ; Data buffer available
005F 7900 F		376	mov r1, #low HOST_CMD_DATA_CNT
0061 7A00		377	mov r2, #00H
0063 020000 F		378	jmp HOST_XFER_ISR_EXIT ; Exit this isr
		379	;
		380	

LOC OBJ LINE SOURCE

```

381          $EJECT
382          ;
0066      383  RECEIVE_HOST_DATA_ISR00:
384          ;
0066 300049 F 385          jnb  HOST_COMMAND, HOST_XFER_ISR_EXIT
0069 200006 F 386          jb   HOST_CMD_PARAM_XFER,
RECEIVE_HOST_DATA_ISR11
006C 200012 F 387          jb   HOST_XFER_ENABLED, RECEIVE_HOST_DATA_ISR22
006F 020000 F 388          jmp  HOST_XFER_ISR_EXIT          ; Exit this isr
389          ;
0072      390  RECEIVE_HOST_DATA_ISR11:
0072 A799      391          mov  @r1, HOST_DATA
0074 09      392          inc  r1
0075 0A      393          inc  r2
0076 120000 F 394          call SETUP_HOST_TIMEOUT
395          ;
0079 BA0236      396          cjne r2, #02H, HOST_XFER_ISR_EXIT
007C C200 F 397          clr  HOST_CMD_PARAM_XFER          ; Got those params
007E 020000 F 398          jmp  HOST_XFER_ISR_EXIT          ; Exit this isr
399          ;
0081      400  RECEIVE_HOST_DATA_ISR22:
401          ;
0081 A800 F 402          mov  r0, HOST_DATA_PTR          ; Get the current pointer
value
0083 A699      403          mov  @r0, HOST_DATA          ; Send the current byte
404          ;
0085 120000 F 405          call DEC_HOST_XFER_COUNT          ; Account for another
byte
406          ;
0088 D50016 F 407          djnz CTI_BUFFER_COUNT, UPDATE_BUFFER_DATA_PTR ;
Acknowledge byte sent
008B 020000 F 408          jmp  HOST_XFER_ISR_EXIT          ; Exit this isr
409          ;
410

```


LOC	OBJ	LINE	SOURCE
		411	\$EJECT
		412	;
008E		413	TRANSMIT_HOST_DATA_ISR:
		414	;
008E	300021	F 415	jnb HOST_COMMAND, HOST_XFER_ISR_EXIT
0091	30001E	F 416	jnb HOST_XFER_ENABLED, HOST_XFER_ISR_EXIT
		417	;
0094	120000	F 418	call DEC_HOST_XFER_COUNT ; Account for another byte
		419	;
0097	D50003	F 420	djnz CTI_BUFFER_COUNT, TRANSMIT_HOST_DATA_ISR00 ; Acknowledge byte sent
009A	020000	F 421	jmp HOST_XFER_ISR_EXIT ; Exit this isr
		422	;
009D		423	TRANSMIT_HOST_DATA_ISR00:
		424	;
009D	A800	F 425	mov r0, HOST_DATA_PTR ; Get the current pointer value
009F	8699	426	mov HOST_DATA, @r0 ; Send the current byte
		427	;
		428	


```

LOC OBJ      LINE  SOURCE
                429      $EJECT
                430      ;
00A1          431  UPDATE_BUFFER_DATA_PTR:
                432      ;
00A1 0500    F   433      inc  HOST_DATA_PTR          ; Point to next buffer location
00A3 120000  F   434      call SETUP_HOST_TIMEOUT
                435      ;
00A6 E500    F   436      mov  a, HOST_DATA_PTR
00A8 B40000  F   437      cjne a, #low STACK, $+3      ; Check for buffer over-run
00AB 4005     438      jc   HOST_XFER_ISR_EXIT          ;
00AD D200    F   439      setb XFER_BUFFER_FULL        ; Data buffer full, Reset
buff ptr
00AF 750000  F   440      mov  HOST_DATA_PTR, #low XFER_DATA_BUFFER
                441      ;
00B2          442  HOST_XFER_ISR_EXIT:
                443      ;
00B2 D000     444      pop  REGISTER_00
00B4 D0D0     445      pop  psw          ; Restore the status word
00B6 D0E0     446      pop  acc          ; Restore the accumulator
                447      ;
00B8 32       448      reti
                449
                450      END
                ;
                ; End of CTI_ISRS.A51
                ;

```

REGISTER BANK(S) USED: 0

ASSEMBLY COMPLETE, NO ERRORS FOUND

MS-DOS MACRO ASSEMBLER A51 V4.4
OBJECT MODULE PLACED IN CTI_MAIN.OBJ
ASSEMBLER INVOKED BY: A51 CTI_MAIN.A51 DEBUG ERRORPRINT(CTI_MAIN.ERR)
NOSYMBOLS NOXREF

LOC	OBJ	LINE	SOURCE
		1	\$PAGEWIDTH (127)
		2	\$PAGELENGTH (57)
		3	;
		4	\$TITLE (CTI_MAIN.A51)
		5	;
		6	;
		7	Program Title: Cellular Telephone Interface Controller Firmwa
		8	Filename : CTI_MAIN.A51
		9	Module Name : CTI_MAIN.OBJ
		10	Project # :
		11	Author : Theodore W. Watler
		12	From : Parchment Designs
		13	For : Turner, Gold, France & Associates
		14	Date Created : August 2, 1991
		15	Version : A.00
		16	;
		17	;
		18	COPYRIGHT (C) 1991. ALL RIGHTS RESERVED
		19	Turner, Gold, France & Associates
		20	;
		21	;
		22	;
		23	PROGRAM FUNCTION
		24	;
		25	;
		26	PROGRAM DESCRIPTION
		27	;
		28	;
		29	REFERENCES
		30	;
		31	1. 8051 Hardware Reference Manual
		32	2. Franklin Software DK51 Development Tools
		33	3.
		34	;
		35	***** MODULE HISTORY *****
		36	;
		37	;
		38	#####

LOC	OBJ	LINE	SOURCE
		39	\$EJECT
		40	;
		41	NAME CTI_MAIN_MODULE
		42	;
		43	EXTERNAL REFERENCE TABLE
		44	;
		45	EXTRN CODE (CHECK_PHONE_STATUS)
		46	EXTRN CODE (RESET_CTI_VARIABLES)
		47	EXTRN CODE (SETUP_HOST_TIMEOUT)
		48	EXTRN CODE (SETUP_TPHONE_TIMEOUT)
		49	
		50	EXTRN CODE (CMD_READ_PTR_PHONE_NUMBER)
		51	EXTRN CODE (CMD_READ_PHONE_CALLS)
		52	EXTRN CODE (CMD_READ_PHONE_TIME)
		53	EXTRN CODE (CMD_WRITE_PHONE_TIME)
		54	EXTRN CODE (CMD_TURN_PHONE_OFF)
		55	EXTRN CODE (CMD_READ_PHONE_RTb_VER)
		56	EXTRN CODE (CMD_READ_NOVATEL_VER)
		57	EXTRN CODE (CMD_READ_CTI_VERSION)
		58	EXTRN CODE (CMD_TURN_POWER_ON)
		59	EXTRN CODE (CMD_LOCK_PHONE)
		60	EXTRN CODE (CMD_UNLOCK_PHONE)
		61	EXTRN CODE (CMD_READ_AIR_TIME_METER)
		62	EXTRN CODE (CMD_FAKE_POWER_DOWN)
		63	EXTRN CODE (CMD_READ_CALLS_COUNTER)
		64	EXTRN CODE (CMD_READ_CALLS_RAM_PTR)
		65	EXTRN CODE (CMD_WRITE_TELEMAC_FIRMWARE)
		66	EXTRN CODE (CMD_PHONE_IN_CRADDLE)
		67	EXTRN CODE (CMD_RESET_CALLS_POINTER)
		68	EXTRN CODE (CMD_RESET_CALLS_COUNTER)
		69	EXTRN CODE (CMD_RESET_AIR_TIME_METER)
		70	
		71	EXTRN IDATA (STACK)
		72	;
		73	PUBLIC DECLARATION TABLE
		74	;
		75	PUBLIC CTI_MAIN_FUNCTION
		76	;
		283	\$LIST
		284	;
		285	
		286	CTI_MAIN_MODULE SEGMENT CODE
----		287	RSEG CTI_MAIN_MODULE
		288	USING REG_BANK_00
		289	


```

LOC OBJ      LINE  SOURCE
                290      $EJECT
                291      ;
                292      ;*<
                293      ;  NAME:
                294      ;
                295      ;  DESCRIPTION:
                296      ;  CALL:
                297      ;  ARGUMENTS:
                298      ;  MODIFIES:
                299      ;  RETURNS:
                300      ;  HISTORY:
                301      ;*>
                302      ;
0000          303  CTI_MAIN_FUNCTION:
                304      ;
0000 C298      305      clr  HOST_RXD          ; Clear for host data
0002 C299      306      clr  HOST_TXD          ; Incase a fake received came in
0004 C200      F  307      clr  HOST_COMMAND      ; Clear host command flag
                308      ;
0006 D200      F  309      setb CTI_PTR_TXD        ; Keep line high to phone
0008 C200      F  310      clr  CTI_HOST_CTS      ; Flag host ready for action!!!
                311      ;
000A 758100    F  312      mov  sp, #low STACK-1    ; Initialize the top of stack
000D 750000    F  313      mov  PTR_PULSE_COUNT, #00H ; Clear the phone
EXCLK pulse cnt
0010 D2A8      314      setb  PTR_800_INTERRUPT    ; Turn on the phone
interrupt
0012 D2AC      315      setb  HOST_COMM_INTERRUPT  ; Turn on the serial
comm interrupt
                316      ;
0014          317  CTI_MAIN_00:
0014 200006    F  318      jb   HOST_TURN_PHONE_ON, CTI_MAIN_11 ; Host turned on
the phone
0017 300503      319      jnb  CTI_TURN_PHONE_OFF, CTI_MAIN_11 ; Inactive so
forget it
001A 120000    F  320      call CTI_KEEP_CTPHONE_OFF
                321      ;
001D          322  CTI_MAIN_11:
001D 2000E0    F  323      jb   CTI_HOST_RTS, CTI_MAIN_FUNCTION ; Host not ready.
Wait!!!
0020 3000F1    F  324      jnb  HOST_COMMAND, CTI_MAIN_00    ; Host online,
Command ???
0023 120000    F  325      call PROCESS_HOST_COMMAND      ; The host current
command
                326      ;
0026          327  CTI_MAIN_22:
0026 120000    F  328      call RESET_CTI_VARIABLES      ; Clean up and return
0029 0100      F  329      ajmp CTI_MAIN_FUNCTION
                330      ;
                331

```


LOC	OBJ	LINE	SOURCE
		332	\$EJECT
		333	;
		334	;* <
		335	NAME:
		336	;
		337	DESCRIPTION:
		338	CALL:
		339	ARGUMENTS:
		340	MODIFIES:
		341	RETURNS:
		342	HISTORY:
		343	;* >
		344	;
002B		345	PROCESS_HOST_COMMAND:
		346	;
002B C298		347	clr HOST_RXD ; Clear for host data
002D C299		348	clr HOST_TXD ; Incase a fake received came in
002F E500 F		349	mov a, HOST_CMD_REG ; Get the current host
			command
0031 B41600		350	cjne a, #CTI_MAX_COMMAND_COUNT, \$+3
0034 5024		351	jnc PROCESS_CMD_EXIT ; Illegal Command quit
0036		352	PROCESS_CMD00:
		353	;
0036 B40403		354	cjne a, #CTI_TBD_04, PROCESS_CMD01
0039 020000 F		355	jmp PROCESS_CMD_EXIT ; Illegal Command quit
003C		356	PROCESS_CMD01:
		357	;
003C B40C03		358	cjne a, #CTI_TBD_12, PROCESS_CMD02
003F 020000 F		359	jmp PROCESS_CMD_EXIT ; Illegal Command quit
0042		360	PROCESS_CMD02:
		361	;
0042 75F000		362	mov b, #00H ; Clear the b register
0045 B41203		363	cjne a, #CTI_PHONE_IN_CRADDLE, PROCESS_CMD03
0048 8500F0 F		364	mov b, HOST_CTI_STATUS
004B		365	PROCESS_CMD03:
		366	;
004B 85F000 F		367	mov HOST_CTI_STATUS, b ; Clear or add previous
			status???
004E C2A8		368	clr PTR_800_INTERRUPT ; Turn off the phone
			interrupt
0050 C205		369	clr CTI_TURN_PHONE_OFF
0052 75F006		370	mov b, #06H ; Byte count for ljmp instruction
0055 A4		371	mul ab ; Multiply by 3 for command table
0056 900000 F		372	mov dptr, #HOST_COMMAND_TABLE
0059 73		373	jmp @a+dptr ; Go execute the command
		374	;
005A		375	PROCESS_CMD_EXIT:
		376	;
005A 22		377	ret
		378	

LOC	OBJ	LINE	SOURCE
		379	\$EJECT
		380	;
		381	;*<
		382	NAME:
		383	;
		384	DESCRIPTION:
		385	CALL:
		386	ARGUMENTS:
		387	MODIFIES:
		388	RETURNS:
		389	HISTORY:
		390	;>
		391	;
005B		392	HOST_COMMAND_TABLE:
		393	;
005B	750006	F 394	mov CTI_BUFFER_COUNT, #06H
005E	020000	F 395	ljmp CMD_READ_PTR_PHONE_NUMBER
		396	;
0061	750038	F 397	mov CTI_BUFFER_COUNT, #38H
0064	020000	F 398	ljmp CMD_READ_PHONE_CALLS
		399	;
0067	750008	F 400	mov CTI_BUFFER_COUNT, #08H
006A	020000	F 401	ljmp CMD_READ_PHONE_TIME
		402	;
006D	750008	F 403	mov CTI_BUFFER_COUNT, #08H
0070	020000	F 404	ljmp CMD_WRITE_PHONE_TIME
		405	;
0073	750000	F 406	mov CTI_BUFFER_COUNT, #00H
0076	020000	F 407	ljmp RESET_CTI_VARIABLES ; TBD 0x04
		408	;
0079	750000	F 409	mov CTI_BUFFER_COUNT, #00H
007C	020000	F 410	ljmp CMD_TURN_PHONE_OFF
		411	;
007F	75000C	F 412	mov CTI_BUFFER_COUNT, #0CH
0082	020000	F 413	ljmp CMD_READ_PHONE_RTb_VER
		414	;
0085	75000C	F 415	mov CTI_BUFFER_COUNT, #0CH
0088	020000	F 416	ljmp CMD_READ_NOVATEL_VER
		417	;
008B	75000C	F 418	mov CTI_BUFFER_COUNT, #0CH
008E	020000	F 419	ljmp CMD_READ_CTI_VERSION
		420	;
0091	750000	F 421	mov CTI_BUFFER_COUNT, #00H
0094	020000	F 422	ljmp CMD_TURN_POWER_ON
		423	;
0097	750000	F 424	mov CTI_BUFFER_COUNT, #00H
009A	020000	F 425	ljmp CMD_LOCK_PHONE
		426	;
009D	750000	F 427	mov CTI_BUFFER_COUNT, #00H
00A0	020000	F 428	ljmp CMD_UNLOCK_PHONE
		429	;

LOC	OBJ	LINE	SOURCE
00A3	750000 F	430	mov CTI_BUFFER_COUNT, #00H
00A6	020000 F	431	ljmp RESET_CTI_VARIABLES ; TBD 0x0C
		432	;
00A9	750008 F	433	mov CTI_BUFFER_COUNT, #08H
00AC	020000 F	434	ljmp CMD_READ_AIR_TIME_METER
		435	;
00AF	750000 F	436	mov CTI_BUFFER_COUNT, #00H
00B2	020000 F	437	ljmp CMD_FAKE_POWER_DOWN
		438	;
00B5	750002 F	439	mov CTI_BUFFER_COUNT, #02H
00B8	020000 F	440	ljmp CMD_READ_CALLS_COUNTER
		441	;
00BB	750002 F	442	mov CTI_BUFFER_COUNT, #02H
00BE	020000 F	443	ljmp CMD_READ_CALLS_RAM_PTR
		444	;
00C1	750038 F	445	mov CTI_BUFFER_COUNT, #38H
00C4	020000 F	446	ljmp CMD_WRITE_TELEMAC_FIRMWARE
		447	;
00C7	750001 F	448	mov CTI_BUFFER_COUNT, #01H
00CA	020000 F	449	ljmp CMD_PHONE_IN_CRADDLE
		450	;
00CD	750000 F	451	mov CTI_BUFFER_COUNT, #00H
00D0	020000 F	452	ljmp CMD_RESET_CALLS_POINTER
		453	;
00D3	750000 F	454	mov CTI_BUFFER_COUNT, #00H
00D6	020000 F	455	ljmp CMD_RESET_CALLS_COUNTER
		456	;
00D9	750000 F	457	mov CTI_BUFFER_COUNT, #00H
00DC	020000 F	458	ljmp CMD_RESET_AIR_TIME_METER
		459	

LOC OBJ LINE SOURCE

```

460          $EJECT
461          ;
462          ;*<
463          ; NAME:
464          ;
465          ; DESCRIPTION:
466          ; CALL:
467          ; ARGUMENTS:
468          ; MODIFIES:
469          ; RETURNS:
470          ; HISTORY:
471          ;*>
472          ;
00DF          473 CTI_KEEP_CTPHONE_OFF:
474          ;
00DF C2A8      475          clr PTR_800_INTERRUPT          ; Turn off the phone
interrupt
00E1 C205      476          clr CTI_TURN_PHONE_OFF
00E3 75000E F  477          mov HOST_CMD_REG,#CTI_FAKE_POWER_DOWN
00E6 120000 F  478          call CMD_FAKE_POWER_DOWN          ; Turn everything off
00E9 120000 F  479          call RESET_CTI_VARIABLES          ; Clean up and return
480          ;
00EC 22        481          ret
482
483          END
          ;
          ; End of CTI_MAIN.A51
          ;

```

REGISTER BANK(S) USED: 0

ASSEMBLY COMPLETE, NO ERRORS FOUND

MS-DOS MACRO ASSEMBLER A51 V4.4
OBJECT MODULE PLACED IN CTI_UTIL.OBJ
ASSEMBLER INVOKED BY: A51 CTI_UTIL.A51 DEBUG ERRORPRINT(CTI_UTIL.ERR)
NOSYMBOLS NOXREF

LOC	OBJ	LINE	SOURCE
		1	\$PAGEWIDTH (127)
		2	\$PAGELENGTH (57)
		3	;
		4	\$TITLE (CTI_UTIL.A51)
		5	;
		6	;
		7	Program Title: Cellular Telephone Interface Controller Firmwa
		8	Filename : CTI_UTIL.A51
		9	Module Name : CTI_UTIL.OBJ
		10	Project # :
		11	Author : Theodore W. Watler
		12	From : Parchment Designs
		13	For : Turner, Gold, France & Associates
		14	Date Created : August 8, 1991
		15	Version : A.00
		16	;
		17	;
		18	COPYRIGHT (C) 1991. ALL RIGHTS RESERVED
		19	Turner, Gold, France & Associates
		20	;
		21	;
		22	;
		23	PROGRAM FUNCTION
		24	;
		25	;
		26	PROGRAM DESCRIPTION
		27	;
		28	;
		29	REFERENCES
		30	;
		31	1. 8051 Hardware Reference Manual
		32	2. Franklin Software DK51 Development Tools
		33	3.
		34	;
		35	***** MODULE HISTORY *****
		36	;
		37	;
		38	#####

LOC	OBJ	LINE	SOURCE
		39	\$EJECT
		40	;
		41	NAME CTI_SUPPORT_FUNCTIONS
		42	;
		43	EXTERNAL REFERENCE TABLE
		44	;
		45	EXTRN CODE (CMD_FAKE_POWER_DOWN)
		46	;
		47	PUBLIC DECLARATION TABLE
		48	;
		49	PUBLIC CHECK_PHONE_STATUS
		50	PUBLIC DEC_HOST_XFER_COUNT
		51	PUBLIC RESET_CTI_VARIABLES
		52	PUBLIC SETUP_HOST_TIMEOUT
		53	PUBLIC SETUP_TPHONE_TIMEOUT
		54	PUBLIC TIME_DELAY
		55	PUBLIC DELAY_350_MSECS
		56	PUBLIC DELAY_WRITE_RAM_FIRMWARE
		57	;
		264	\$LIST
		265	;
		266	
		267	CTI_UTILITIES SEGMENT CODE
----		268	RSEG CTI_UTILITIES
		269	USING REG_BANK_00
		270	


```

LOC OBJ      LINE  SOURCE
                271      $EJECT
                272      ;
                273      ;*<
                274      ;  NAME:
                275      ;
                276      ;  DESCRIPTION:
                277      ;  CALL:
                278      ;  ARGUMENTS:
                279      ;  MODIFIES:
                280      ;  RETURNS:
                281      ;  HISTORY:
                282      ;*>
                283      ;
0000          284  RESET_CTI_VARIABLES:
                285      ;
0000 A200      F    286      mov  c, HOST_XFER_TIMEDOUT
0002 9200      F    287      mov  CTI_HOST_TIMEDOUT, c
0004 A200      F    288      mov  c, PTR_XFER_TIMEDOUT
0006 9200      F    289      mov  CTI_CTPHONE_TIMEDOUT, c
0008 A200      F    290      mov  c, LIVE_PHONE_IN_CRADDLE
000A 9200      F    291      mov  CTI_ACTIVE_CTPHONE, c
                292      ;
000C 300506    293      jnb  CTI_TURN_PHONE_OFF, RESET_CTI_VARIABLES00
000F 75000E    F    294      mov  HOST_CMD_REG, #CTI_FAKE_POWER_DOWN
0012 120000    F    295      call CMD_FAKE_POWER_DOWN
                296      ;
0015          297  RESET_CTI_VARIABLES00:
0015 C298      298      clr  HOST_RXD          ; Clear host received bit
0017 C299      299      clr  HOST_TXD          ; Clear host transmit bit
0019 7500FF    F    300      mov  HOST_CMD_REG, #0FFH      ; Invalid command
001C C200      F    301      clr  HOST_XFER_ENABLED
001E C200      F    302      clr  HOST_XFER_TIMEDOUT      ; Clear host timed out
flag
0020 C200      F    303      clr  HOST_CMD_PARAM_XFER      ; Clear Host param flag
0022 750000    F    304      mov  HOST_DATA_PTR, #low XFER_DATA_BUFFER
0025 750000    F    305      mov  HOST_XFER_COUNT, #00H      ; Clear the host byte
count lsb
0028 750000    F    306      mov  HOST_XFER_COUNT + 1, #00H  ; Clear the host byte
count msb
                307      ;
002B C200      F    308      clr  PTR_XFER_ENABLED
002D C200      F    309      clr  PTR_XFER_TIMEDOUT
002F D2A8      310      setb  PTR_800_INTERRUPT
0031 750000    F    311      mov  PTR_PULSE_COUNT, #00H
                312      ;
0034 C200      F    313      clr  CTI_HOST_CTS          ; Flag host ready for action!!!
0036 C205      314      clr  CTI_TURN_PHONE_OFF
0038 750000    F    315      mov  CTI_BUFFER_COUNT, #00H      ; Clear the current fifo
count
003B C200      F    316      clr  XFER_BUFFER_FULL      ; Data buffer available
                317      ;
003D C2A9      318      clr  XFER_TIMEOUT_INTERRUPT
003F C28C      319      clr  START_CTI_TIMEOUT      ; Stop the timeout counter
0041 758841    320      mov  tcon, #01000001B      ; Set timer control to all bits off

```


321 . ;

431

LOC	OBJ	LINE	SOURCE
0044	7800	F 322	mov r0, #low XFER_DATA_BUFFER
0046	7938	323	mov r1, #MAX_BUFFER_SIZE
		324	;
		325	; Clear the transfer buffer RAM
		326	;
0048		327	RESET_CTI_VARIABLES11:
0048	7600	328	mov @r0, #00H
004A	08	329	inc r0
004B	D9FB	330	djnz r1, RESET_CTI_VARIABLES11
		331	;
004D	22	332	ret
		333	

LOC	OBJ	LINE	SOURCE
		334	\$EJECT
		335	;
		336	;*<
		337	; NAME:
		338	;
		339	; DESCRIPTION:
		340	; CALL:
		341	; ARGUMENTS:
		342	; MODIFIES:
		343	; RETURNS:
		344	; HISTORY:
		345	;*>
		346	;
004E		347	CHECK_PHONE_STATUS:
		348	;
004E D200	F	349	setb PTR_ONLINE_CHECK
0050 D200	F	350	setb PTR_XFER_ENABLED
0052 D2A8		351	setb PTR_800_INTERRUPT
0054 750000	F	352	mov PTR_PULSE_COUNT, #00H
0057 C200	F	353	clr LIVE_PHONE_IN_CRADDLE
		354	;
0059 750001	F	355	mov DELAY_CTR_02, #01H
005C 75007E	F	356	mov DELAY_CTR_01, #7EH
005F 120000	F	357	call TIME_DELAY ; Wait for ~70msecs
		358	;
0062 7D00		359	mov r5, #000H
0064 7E4C		360	mov r6, #04CH
0066 120000	F	361	call SETUP_TPHONE_TIMEOUT ; Time out in ~70msecs
0069 D2A9		362	setb XFER_TIMEOUT_INTERRUPT
		363	;
006B		364	CHECK_PHONE_STATUS00:
006B 200006	F	365	jb PTR_XFER_TIMEDOUT, CHECK_PHONE_STATUS11
006E 3000FA	F	366	jnb LIVE_PHONE_IN_CRADDLE,
			CHECK_PHONE_STATUS00
0071 200002	F	367	jb LIVE_PHONE_IN_CRADDLE,
			CHECK_PHONE_STATUS_EXIT
		368	;
0074		369	CHECK_PHONE_STATUS11:
0074 D205		370	setb CTI_TURN_PHONE_OFF
		371	;
0076		372	CHECK_PHONE_STATUS_EXIT:
0076 C2A9		373	clr XFER_TIMEOUT_INTERRUPT
0078 C2A8		374	clr PTR_800_INTERRUPT
007A C200	F	375	clr PTR_XFER_ENABLED
007C C200	F	376	clr PTR_ONLINE_CHECK
		377	;
007E 22		378	ret
		379	

LOC	OBJ	LINE	SOURCE
		380	\$EJECT
		381	;
		382	;*<
		383	; NAME:
		384	;
		385	; DESCRIPTION:
		386	; CALL:
		387	; ARGUMENTS:
		388	; MODIFIES:
		389	; RETURNS:
		390	; HISTORY:
		391	;*>
		392	;
007F		393	DEC_HOST_XFER_COUNT:
		394	;
007F C0E0		395	push acc ; Save the accumulator
		396	;
0081 C3		397	clr c
0082 E500	F	398	mov a, HOST_XFER_COUNT ;
0084 9401		399	subb a, #01H ; Another byte xfered
0086 C500	F	400	xch a, HOST_XFER_COUNT ;
0088 E500	F	401	mov a, HOST_XFER_COUNT + 1
008A 9400		402	subb a, #00H
008C F500	F	403	mov HOST_XFER_COUNT + 1, a
		404	;
008E 4500	F	405	orl a, HOST_XFER_COUNT ; If transfer complete???
0090 7002		406	jnz DEC_HOST_XFER_COUNT_EXIT ; Quit if not else
0092 D200	F	407	setb HOST_XFER_COMPLETE ; Flag it
		408	;
0094		409	DEC_HOST_XFER_COUNT_EXIT:
0094 D0E0		410	pop acc ; Restore the accumulator
0096 22		411	ret
		412	

LOC	OBJ	LINE	SOURCE
		413	\$EJECT
		414	;
		415	;*<
		416	; NAME:
		417	;
		418	; DESCRIPTION:
		419	; CALL:
		420	; ARGUMENTS:
		421	; MODIFIES:
		422	; RETURNS:
		423	; HISTORY:
		424	;*>
		425	;
0097		426	SETUP_HOST_TIMEOUT:
		427	;
0097	C28C	428	clr START_CTI_TIMEOUT ; Stop the timeout counter
0099	758AC0	429	mov dl0, #low HOST_TIMEOUT_CNT ; Reset timeout counter
009C	758CF4	430	mov th0, #high HOST_TIMEOUT_CNT
009F	758841	431	mov tcon, #01000001B ; Set timer control to all bits off
00A2	D28C	432	setb START_CTI_TIMEOUT ; Restart the timeout counter
		433	;
00A4	C200	F 434	clr HOST_XFER_TIMEDOUT
		435	;
00A6	22	436	ret
		437	

LOC OBJ LINE SOURCE

```

438          $EJECT
439      ;
440      ;*<
441      ; NAME:
442      ;
443      ; DESCRIPTION:
444      ; CALL:
445      ; ARGUMENTS:
446      ; MODIFIES:
447      ; RETURNS:
448      ; HISTORY:
449      ;*>
450      ;
00A7      451  SETUP_TPHONE_TIMEOUT:
452      ;
00A7 C28C      453      clr  START_CTI_TIMEOUT      ; Stop the timeout counter
00A9 8D8A      454      mov  tl0, r5      ; Reset timeout counter
00AB 8E8C      455      mov  th0, r6
00AD 758841     456      mov  tcon, #01000001B      ; Set timer control to all bits off
00B0 D28C      457      setb  START_CTI_TIMEOUT      ; Restart the timeout
counter
458      ;
00B2 C200      F 459      clr  PTR_XFER_TIMEDOUT
460      ;
00B4 22        461      ret
462

```


LOC OBJ LINE SOURCE

```

463          $EJECT
464          ;
465          ;*<
466          ; NAME: Time_Delay
467          ;
468          ; DESCRIPTION:
469          ; CALL:
470          ; ARGUMENTS:
471          ; MODIFIES:
472          ; RETURNS:
473          ; HISTORY:
474          ;*>
475          ;
00B5          476 TIME_DELAY:
00B5 750000 F 477          mov  DELAY_CTR_00, #000H          ; Clear 553.8542 usec
counter
478          ;
00B8          479 TIME_DELAY00:
00B8 D500FD F 480          djnz  DELAY_CTR_00, $          ; Count down for 553.8542
us
00BB D500FA F 481          djnz  DELAY_CTR_01, TIME_DELAY00
00BE D500F7 F 482          djnz  DELAY_CTR_02, TIME_DELAY00
483          ;
00C1 22          484          ret
485

```


LOC OBJ LINE SOURCE

```

      486          $EJECT
      487      ;
      488      ;*<
      489      ;   NAME:
      490      ;
      491      ;   DESCRIPTION:
      492      ;   CALL:
      493      ;   ARGUMENTS:
      494      ;   MODIFIES:
      495      ;   RETURNS:
      496      ;   HISTORY:
      497      ;*>
      498      ;
00C2      499      DELAY_350_MSECS:
      500      ;
00C2 750003 F      501          mov    DELAY_CTR_02, #03H          ; Setup 350msec delay
count
00C5 750078 F      502          mov    DELAY_CTR_01, #78H
00C8 120000 F      503          call   TIME_DELAY          ; Go and wait for that time
      504      ;
00CB 22      505          ret
      506
```


LOC	OBJ	LINE	SOURCE
		507	\$EJECT
		508	;
		509	;*<
		510	; NAME:
		511	;
		512	; DESCRIPTION:
		513	; CALL:
		514	; ARGUMENTS:
		515	; MODIFIES:
		516	; RETURNS:
		517	; HISTORY:
		518	;*>
		519	;
00CC		520	DELAY_WRITE_RAM_FIRMWARE:
		521	;
00CC 750001	F	522	mov DELAY_CTR_02, #01H ; Setup 350msec delay
count			
00CF 75007E	F	523	mov DELAY_CTR_01, #7EH
00D2 120000	F	524	call TIME_DELAY ; Go and wait for that time
		525	;
00D5 22		526	ret
		527	
		528	END
			;
			; End of CTI_UTIL.A51
			;

REGISTER BANK(S) USED: 0

ASSEMBLY COMPLETE, NO ERRORS FOUND

LOC OBJ LINE SOURCE

```

    507          $EJECT
    508          ;
    509          ;*<
    510          ;   NAME:
    511          ;
    512          ;   DESCRIPTION:
    513          ;   CALL:
    514          ;   ARGUMENTS:
    515          ;   MODIFIES:
    516          ;   RETURNS:
    517          ;   HISTORY:
    518          ;*>
    519          ;
00CC          520  DELAY_WRITE_RAM_FIRMWARE:
    521          ;
00CC 750001 F   522          mov  DELAY_CTR_02, #01H          ; Setup 350msec delay
count
00CF 75007E F   523          mov  DELAY_CTR_01, #7EH
00D2 120000 F   524          call TIME_DELAY          ; Go and wait for that time
    525          ;
00D5 22          526          ret
    527
    528          END
          ;
          ; End of CTI_UTIL.A51
          ;
```

REGISTER BANK(S) USED: 0

ASSEMBLY COMPLETE, NO ERRORS FOUND

MS-DOS MACRO ASSEMBLER A51 V4.4
OBJECT MODULE PLACED IN CTI_VARS.OBJ
ASSEMBLER INVOKED BY: A51 CTI_VARS.INC

LOC	OBJ	LINE	SOURCE
		1 ;	\$PAGEWIDTH (127)
		2 ;	\$PAGELENGTH (57)
		3 ;	
		4 ;	\$TITLE (CTI_VARS.INC)
		5 ;	
		6 ;	Program Title: Cellular Telephone Interface Controller Firmwa
		7 ;	Filename : CTI_VARS.INC
		8 ;	Project # :
		9 ;	Author : Theodore W. Watler
		10 ;	From : Parchment Designs
		11 ;	For : Turner, Gold, France & Associates
		12 ;	Date Created : August 4, 1991
		13 ;	Version : A.00
		14 ;	
		15 ;	
		16 ;	
		17 ;	COPYRIGHT (C) 1991. ALL RIGHTS RESERVED
		18 ;	Turner, Gold, France & Associates
		19 ;	
		20 ;	
		21 ;	
		22 ;	PROGRAM FUNCTION
		23 ;	
		24 ;	
		25 ;	PROGRAM DESCRIPTION
		26 ;	
		27 ;	
		28 ;	REFERENCES
		29 ;	
		30 ;	1. 8051 Hardware Reference Manual
		31 ;	2. Franklin Software DK51 Development Tools
		32 ;	3.
		33 ;	
		34 ;	***** MODULE HISTORY *****
		35 ;	
		36	
		37	

#####

LOC	OBJ	LINE	SOURCE
		38	\$EJECT
		39	;
		40	;
		41	EXTERNALS DECLARATION TABLE
		42	;
		43	EXTRN BIT(CTI_HOST_RTS)
		44	EXTRN BIT(CTI_HOST_CTS)
		45	EXTRN BIT(CTI_PTR_RXD)
		46	EXTRN BIT(CTI_PTR_EXPWR_ON)
		47	EXTRN BIT(CTI_PTR_TXD)
		48	EXTRN BIT(CTI_LED_1)
		49	EXTRN BIT(CTI_LED_2)
		50	EXTRN BIT(CTI_PTR_EXCLK)
		51	EXTRN BIT(CTI_HOST_TXD)
		52	EXTRN BIT(CTI_HOST_RXD)
		53	
		54	EXTRN DATA(DELAY_CTR_02)
		55	EXTRN DATA(DELAY_CTR_01)
		56	EXTRN DATA(DELAY_CTR_00)
		57	
		58	EXTRN DATA(HOST_CMD_REG)
		59	EXTRN DATA(HOST_CMD_DATA_CNT)
		60	EXTRN DATA(HOST_DATA_PTR)
		61	EXTRN DATA(HOST_XFER_COUNT)
		62	EXTRN DATA(PTR_BIT_COUNT)
		63	EXTRN DATA(PTR_PULSE_COUNT)
		64	EXTRN DATA(PTR_CHECKSUM_REG)
		65	EXTRN DATA(CTI_BUFFER_COUNT)
		66	;
		67	EXTRN DATA(CTI_STATUS)
		68	EXTRN BIT(CTI_TURN_OFF_PHONE)
		69	EXTRN BIT(LIVE_PHONE_IN_CRADDLE)
		70	EXTRN BIT(HOST_TURN_PHONE_ON)
		71	EXTRN BIT(XFER_BUFFER_FULL)
		72	EXTRN BIT(HOST_COMMAND)
		73	;
		74	EXTRN DATA(HOST_CTI_STATUS)
		75	EXTRN BIT(CTI_HOST_TIMEDOUT)
		76	EXTRN BIT(CTI_CTPHONE_TIMEDOUT)
		77	EXTRN BIT(CTI_ACTIVE_CTPHONE)
		78	;
		79	EXTRN DATA(HOST_XFER_STATUS)
		80	EXTRN BIT(HOST_CMD_PARAM_XFER)
		81	EXTRN BIT(HOST_XFER_COMPLETE)
		82	EXTRN BIT(HOST_XFER_TIMEDOUT)
		83	EXTRN BIT(HOST_XFER_ENABLED)
		84	;
		85	EXTRN DATA(PTR_XFER_STATUS)
		86	EXTRN BIT(PTR_EDGE_DETECTED)
		87	EXTRN BIT(PTR_ONLINE_CHECK)
		88	EXTRN BIT(PTR_DATA_TRANSMITTED)
		89	EXTRN BIT(PTR_XFER_TIMEDOUT)
		90	EXTRN BIT(PTR_XFER_ENABLED)
		91	;


```
92      EXTRN DATA(XFER_DATA_BUFFER)
93  ;
94  ; End of CTI_VARS.INC
95  ;
96
```


SYMBOL TABLE LISTING

NAME	TYPE	VALUE	ATTRIBUTES
CTI_ACTIVE_CTPHONE..	B ADDR	----	EXT
CTI_BUFFER_COUNT...	D ADDR	----	EXT
CTI_CTPHONE_TIMEDOUT.	B ADDR	----	EXT
CTI_HOST_CTS.....	B ADDR	----	EXT
CTI_HOST_RTS.....	B ADDR	----	EXT
CTI_HOST_RXD.....	B ADDR	----	EXT
CTI_HOST_TIMEDOUT..	B ADDR	----	EXT
CTI_HOST_TXD.....	B ADDR	----	EXT
CTI_LED_1.....	B ADDR	----	EXT
CTI_LED_2.....	B ADDR	----	EXT
CTI_PTR_EXCLK....	B ADDR	----	EXT
CTI_PTR_EXPWR_ON...	B ADDR	----	EXT
CTI_PTR_RXD.....	B ADDR	----	EXT
CTI_PTR_TXD.....	B ADDR	----	EXT
CTI_STATUS.....	D ADDR	----	EXT
CTI_TURN_OFF_PHONE..	B ADDR	----	EXT
DELAY_CTR_00.....	D ADDR	----	EXT
DELAY_CTR_01.....	D ADDR	----	EXT
DELAY_CTR_02.....	D ADDR	----	EXT
HOST_CMD_DATA_CNT..	D ADDR	----	EXT
HOST_CMD_PARAM_XFER.	B ADDR	----	EXT
HOST_CMD_REG.....	D ADDR	----	EXT
HOST_COMMAND.....	B ADDR	----	EXT
HOST_CTI_STATUS...	D ADDR	----	EXT
HOST_DATA_PTR....	D ADDR	----	EXT
HOST_TURN_PHONE_ON..	B ADDR	----	EXT
HOST_XFER_COMPLETE..	B ADDR	----	EXT
HOST_XFER_COUNT...	D ADDR	----	EXT
HOST_XFER_ENABLED..	B ADDR	----	EXT
HOST_XFER_STATUS...	D ADDR	----	EXT
HOST_XFER_TIMEDOUT..	B ADDR	----	EXT
LIVE_PHONE_IN_CRADDLE	B ADDR	----	EXT
PTR_BIT_COUNT....	D ADDR	----	EXT
PTR_CHECKSUM_REG...	D ADDR	----	EXT
PTR_DATA_TRANSMITTED.	B ADDR	----	EXT
PTR_EDGE_DETECTED..	B ADDR	----	EXT
PTR_ONLINE_CHECK...	B ADDR	----	EXT
PTR_PULSE_COUNT...	D ADDR	----	EXT
PTR_XFER_ENABLED...	B ADDR	----	EXT
PTR_XFER_STATUS...	D ADDR	----	EXT
PTR_XFER_TIMEDOUT..	B ADDR	----	EXT
XFER_BUFFER_FULL...	B ADDR	----	EXT
XFER_DATA_BUFFER...	D ADDR	----	EXT

REGISTER BANK(S) USED: 0

ASSEMBLY COMPLETE, NO ERRORS FOUND

444

MS-DOS MACRO ASSEMBLER A51 V4.4
OBJECT MODULE PLACED IN CTI_XFER.OBJ
ASSEMBLER INVOKED BY: A51 CTI_XFER.A51 DEBUG ERRORPRINT(CTI_XFER.ERR)
NOSYMBOLS NOXREF

LOC	OBJ	LINE	SOURCE
		1	\$PAGEWIDTH (127)
		2	\$PAGELENGTH (57)
		3	;
		4	\$TITLE (CTI_XFER.A51)
		5	;
		6	;
		7	Program Title: Cellular Telephone Interface Controller Firmwa
		8	Filename : CTI_XFER.A51
		9	Module Name : CTI_XFER.OBJ
		10	Project # :
		11	Author : Theodore W. Watler
		12	From : Parchment Designs
		13	For : Turner, Gold, France & Associates
		14	Date Created : August 7, 1991
		15	Version : A.00
		16	;
		17	;
		18	COPYRIGHT (C) 1991. ALL RIGHTS RESERVED
		19	Turner, Gold, France & Associates
		20	;
		21	;
		22	;
		23	PROGRAM FUNCTION
		24	;
		25	;
		26	PROGRAM DESCRIPTION
		27	;
		28	;
		29	REFERENCES
		30	;
		31	1. 8051 Hardware Reference Manual
		32	2. Franklin Software DK51 Development Tools
		33	3.
		34	;
		35	***** MODULE HISTORY *****
		36	;
		37	;
		38	#####

445

LOC	OBJ	LINE	SOURCE
		39	\$EJECT
		40	;
		41	NAME CTI_COMMUNICATIONS_HANDLER
		42	;
		43	EXTERNAL REFERENCE TABLE
		44	;
		45	EXTRN CODE (SETUP_HOST_TIMEOUT)
		46	EXTRN CODE (SETUP_TPHONE_TIMEOUT)
		47	;
		48	PUBLIC DECLARATION TABLE
		49	;
		50	PUBLIC RECEIVE_HOST_DATA
		51	PUBLIC TRANSFER_HOST_DATA
		52	
		53	PUBLIC RECEIVE_PHONE_DATA
		54	PUBLIC TRANSFER_PHONE_DATA
		55	;
		262	\$LIST
		263	;
		264	CTI_COMMUNICATIONS SEGMENT CODE
----		265	RSEG CTI_COMMUNICATIONS
		266	USING REG_BANK_00
		267	

446

LOC	OBJ	LINE	SOURCE
		268	\$EJECT
		269	;
		270	;*<
		271	; NAME:
		272	;
		273	; DESCRIPTION:
		274	; CALL:
		275	; ARGUMENTS:
		276	; MODIFIES:
		277	; RETURNS:
		278	; HISTORY:
		279	;*>
		280	;
		281	;
0000		282	RECEIVE_HOST_DATA:
		283	;
0000	C298	284	clr HOST_RXD ; Clear receive flag
0002	C299	285	clr HOST_TXD ; Clear transmit flag
0004	C200 F	286	clr CTI_HOST_CTS ; Flag host ready for action!!!
0006	D200 F	287	setb HOST_XFER_ENABLED
		288	;
0008	120000 F	289	call SETUP_HOST_TIMEOUT
000B	D2A9	290	setb XFER_TIMEOUT_INTERRUPT
		291	;
000D		292	RECEIVE_HOST_DATA00:
000D	200004 F	293	jb HOST_XFER_TIMEDOUT, RECEIVE_HOST_DATA11
0010	E500 F	294	mov a, CTI_BUFFER_COUNT
0012	70F9	295	jnz RECEIVE_HOST_DATA00
		296	;
0014		297	RECEIVE_HOST_DATA11:
0014	C2A9	298	clr XFER_TIMEOUT_INTERRUPT
0016	C200 F	299	clr HOST_XFER_ENABLED
0018	C28C	300	clr START_CTI_TIMEOUT
001A	D200 F	301	setb CTI_HOST_CTS ; Stop all host xfers
		302	;
001C	22	303	ret
		304	

447

LOC	OBJ	LINE	SOURCE
		305	\$EJECT
		306	;
		307	;*<
		308	; NAME:
		309	;
		310	; DESCRIPTION:
		311	; CALL:
		312	; ARGUMENTS:
		313	; MODIFIES:
		314	; RETURNS:
		315	; HISTORY:
		316	;*>
		317	;
		318	;
001D		319	TRANSFER_HOST_DATA:
		320	;
001D C298		321	clr HOST_RXD ; Clear receive flag
001F C299		322	clr HOST_TXD ; Clear transmit flag
0021 C200 F		323	clr CTI_HOST_CTS ; Flag host ready for action!!!
0023 D200 F		324	setb HOST_XFER_ENABLED
		325	;
0025 120000 F		326	call SETUP_HOST_TIMEOUT
0028 D2A9		327	setb XFER_TIMEOUT_INTERRUPT
		328	;
002A A800 F		329	mov r0, HOST_DATA_PTR ; Get the current pointer value
002C 8699		330	mov HOST_DATA, @r0 ; Send the current byte
002E 0500 F		331	inc HOST_DATA_PTR ; Point to next available record
		332	;
0030		333	TRANSFER_HOST_DATA00:
0030 200004 F		334	jb HOST_XFER_TIMEDOUT, TRANSFER_HOST_DATA11
0033 E500 F		335	mov a, CTI_BUFFER_COUNT
0035 70F9		336	jnz TRANSFER_HOST_DATA00
		337	;
0037		338	TRANSFER_HOST_DATA11:
0037 D200 F		339	setb CTI_HOST_CTS ; Stop all host xfers
0039 C2A9		340	clr XFER_TIMEOUT_INTERRUPT
003B C200 F		341	clr HOST_XFER_ENABLED
003D C28C		342	clr START_CTI_TIMEOUT
		343	;
003F 22		344	ret
		345	

448

```

LOC OBJ      LINE  SOURCE
                346      $EJECT
                347      ;
                348      ;*<
                349      ;  NAME:
                350      ;
                351      ;  DESCRIPTION:
                352      ;  CALL:
                353      ;  ARGUMENTS:
                354      ;  MODIFIES:
                355      ;  RETURNS:
                356      ;  HISTORY:
                357      ;*>
                358      ;
                359      ;
0040          360  RECEIVE_PHONE_DATA:
                361      ;
0040 E4        362          clr  a                ; No data available
0041 D3        363          setb c                ; Set for start bit
0042 750008 F   364          mov  PTR_BIT_COUNT,#08H        ; Start + Data bits count
0045 D200 F     365          setb PTR_XFER_ENABLED
0047 D2A9      366          setb XFER_TIMEOUT_INTERRUPT
                367      ;
                368      ;  Setup for PTR start bit wait for ~497.77 msecs
                369      ;
0049 7C07      370          mov  r4,#07H                ; Count for ~497.77 msecs wait time
                371      ;
004B          372  RECV_PHONE_DATA00:                ; Setup for start bit wait
004B 7D00      373          mov  r5,#00H
004D 7E00      374          mov  r6,#00H
004F 120000 F   375          call SETUP_TPHONE_TIMEOUT
                376      ;
0052          377  RECV_PHONE_DATA11:                ; Wait for the start bit
0052 30000B F   378          jnb  CTI_PTR_RXD,RECV_PHONE_DATA33
0055 200003 F   379          jb   PTR_XFER_TIMEDOUT,RECV_PHONE_DATA22
0058 2000F7 F   380          jb   CTI_PTR_RXD,RECV_PHONE_DATA11
                381      ;
005B          382  RECV_PHONE_DATA22:
005B DCEE      383          djnz  r4,RECV_PHONE_DATA00
005D 20001B F   384          jb   PTR_XFER_TIMEDOUT,RECV_PHONE_DATA_EXIT
                385      ;
                386      ;  If all is well receive the PTR synchronous data
                387      ;
0060          388  RECV_PHONE_DATA33:
0060 120000 F   389          call PTR_RXD_CLOCK        ; Wait for start
0063 200015 F   390          jb   PTR_XFER_TIMEDOUT,RECV_PHONE_DATA_EXIT
0066 120000 F   391          call PTR_RXD_CLOCK        ; Clock for the current bit
xfer
0069 20000F F   392          jb   PTR_XFER_TIMEDOUT,RECV_PHONE_DATA_EXIT
                393      ;
006C          394  RECV_PHONE_DATA44:
006C A200 F     395          mov  c,CTI_PTR_RXD        ; Send the current bit
006E 13        396          rrc  a

```


449

LOC OBJ LINE SOURCE

```
006F 120000 F 397      call PTR_RXD_CLOCK      ; Clokc for the current bit
xfer
0072 200006 F 398      jb  PTR_XFER_TIMEOUT, RECV_PHONE_DATA_EXIT
0075 D500F4 F 399      djnz PTR_BIT_COUNT, RECV_PHONE_DATA44
                        400 ;
                        401 ; Wait one external clock cycle for PTR stop bit
                        402 ;
0078 120000 F 403      call PTR_RXD_CLOCK      ; Clokc for the current bit
xfer
                        404 ;
007B      405 RECV_PHONE_DATA_EXIT:
007B C2A9      406      clr  XFER_TIMEOUT_INTERRUPT
007D C200 F 407      clr  PTR_XFER_ENABLED
007F C28C      408      clr  START_CTI_TIMEOUT
0081 22      409      ret
                        410
```



```

LOC OBJ      LINE  SOURCE
              411      $EJECT
              412      ;
              413      ;*<
              414      ; NAME:
              415      ;
              416      ; DESCRIPTION:
              417      ; CALL:
              418      ; ARGUMENTS:
              419      ; MODIFIES:
              420      ; RETURNS:
              421      ; HISTORY:
              422      ;*>
              423      ;
              424      ;
0082          425  TRANSFER_PHONE_DATA:
              426      ;
0082 D200     F    427          setb  PTR_XFER_ENABLED
0084 750009   F    428          mov   PTR_BIT_COUNT, #09H      ; Start + Data bits count
              429      ;
0087 7D00     430          mov   r5, #low PTR_EXCLK_TIMEOUT_CNT
0089 7EEE     431          mov   r6, #high PTR_EXCLK_TIMEOUT_CNT
008B 120000   F    432          call  SETUP_TPHONE_TIMEOUT
008E C200     F    433          clr   PTR_EDGE_DETECTED
0090 D2A8     434          setb  PTR_800_INTERRUPT
0092 D2A9     435          setb  XFER_TIMEOUT_INTERRUPT
              436      ;
0094          437  XFER_PHONE_DATA00:
0094 200021   F    438          jb   PTR_XFER_TIMEDOUT, XFER_PHONE_DATA_EXIT
0097 3000FA   F    439          jnb  PTR_EDGE_DETECTED, XFER_PHONE_DATA00
              440      ;
009A C2A9     441          clr   XFER_TIMEOUT_INTERRUPT
009C C2A8     442          clr   PTR_800_INTERRUPT
009E C28C     443          clr   START_CTI_TIMEOUT
              444      ;
00A0 120000   F    445          call  PTR_TXD_CLOCK      ; For positive edge sync
00A3 200012   F    446          jb   PTR_XFER_TIMEDOUT, XFER_PHONE_DATA_EXIT
00A6 C3       447          clr   c      ; Clear for start bit
              448      ;
00A7          449  XFER_PHONE_DATA11:
00A7 9200     F    450          mov   CTI_PTR_TXD, c      ; Send the current bit
00A9 120000   F    451          call  PTR_TXD_CLOCK      ; Clock for the current bit
xfer
00AC 200009   F    452          jb   PTR_XFER_TIMEDOUT, XFER_PHONE_DATA_EXIT
00AF 13       453          rrc   a
00B0 D500F4   F    454          djnz  PTR_BIT_COUNT, XFER_PHONE_DATA11
              455      ;
00B3 D200     F    456          setb  CTI_PTR_TXD      ; Set stop bit
00B5 120000   F    457          call  PTR_TXD_CLOCK      ; Clock for the current bit
xfer
              458      ;
00B8          459  XFER_PHONE_DATA_EXIT:
              460      ;
00B8 C200     F    461          clr   PTR_XFER_ENABLED

```


457

LOC OBJ	LINE	SOURCE
00BA C28C	462	clr START_CTI_TIMEOUT
00BC C2A8	463	clr PTR_800_INTERRUPT
00BE C2A9	464	clr XFER_TIMEOUT_INTERRUPT
00C0 C3	465	clr c ; Clear for exit
	466	;
00C1 22	467	ret
	468	

LOC	OBJ	LINE	SOURCE
		469	\$EJECT
		470	;
		471	;*<
		472	; NAME:
		473	;
		474	; DESCRIPTION:
		475	; CALL:
		476	; ARGUMENTS:
		477	; MODIFIES:
		478	; RETURNS:
		479	; HISTORY:
		480	;*>
		481	;
		482	;
00C2		483	PTR_RXD_CLOCK:
		484	;
00C2 7D00		485	mov r5, #low PTR_EXCLK_TIMEOUT_CNT
00C4 7EEE		486	mov r6, #high PTR_EXCLK_TIMEOUT_CNT
00C6 120000 F		487	call SETUP_TPHONE_TIMEOUT
00C9 D2A9		488	setb XFER_TIMEOUT_INTERRUPT
		489	;
00CB		490	PTR_RXD_CLOCK00:
00CB 200014 F		491	jb PTR_XFER_TIMEDOUT, PTR_RXD_CLOCK_EXIT
00CE 3000FA F		492	jnb CTI_PTR_EXCLK, PTR_RXD_CLOCK00 ; Check for
			clock LO
		493	;
00D1 C2A9		494	clr XFER_TIMEOUT_INTERRUPT
00D3 7D00		495	mov r5, #low PTR_EXCLK_TIMEOUT_CNT
00D5 7EEE		496	mov r6, #high PTR_EXCLK_TIMEOUT_CNT
00D7 120000 F		497	call SETUP_TPHONE_TIMEOUT
00DA D2A9		498	setb XFER_TIMEOUT_INTERRUPT
		499	;
00DC		500	PTR_RXD_CLOCK11:
00DC 200003 F		501	jb PTR_XFER_TIMEDOUT, PTR_RXD_CLOCK_EXIT
00DF 2000FA F		502	jb CTI_PTR_EXCLK, PTR_RXD_CLOCK11 ; Check for
			clock HI
		503	;
00E2		504	PTR_RXD_CLOCK_EXIT:
		505	;
00E2 C2A9		506	clr XFER_TIMEOUT_INTERRUPT
00E4 C28C		507	clr START_CTI_TIMEOUT
00E6 22		508	ret
		509	


```

LOC OBJ      LINE  SOURCE
                510      $EJECT
                511      ;
                512      ;*<
                513      ; NAME:
                514      ;
                515      ; DESCRIPTION:
                516      ; CALL:
                517      ; ARGUMENTS:
                518      ; MODIFIES:
                519      ; RETURNS:
                520      ; HISTORY:
                521      ;*>
                522      ;
                523      ;
00E7          524  PTR_TXD_CLOCK:
                525      ;
00E7 7D00      526      mov  r5, #low PTR_EXCLK_TIMEOUT_CNT
00E9 7EEE      527      mov  r6, #high PTR_EXCLK_TIMEOUT_CNT
00EB 120000 F  528      call SETUP_TPHONE_TIMEOUT
00EE D2A9      529      setb XFER_TIMEOUT_INTERRUPT
                530      ;
00F0          531  PTR_TXD_CLOCK00:
00F0 200014 F  532      jb   PTR_XFER_TIMEDOUT, PTR_TXD_CLOCK_EXIT
00F3 2000FA F  533      jb   CTI_PTR_EXCLK, PTR_TXD_CLOCK00 ; Check for
clock HI
                534      ;
00F6 C2A9      535      clr  XFER_TIMEOUT_INTERRUPT
00F8 7D00      536      mov  r5, #low PTR_EXCLK_TIMEOUT_CNT
00FA 7EEE      537      mov  r6, #high PTR_EXCLK_TIMEOUT_CNT
00FC 120000 F  538      call SETUP_TPHONE_TIMEOUT
00FF D2A9      539      setb XFER_TIMEOUT_INTERRUPT
                540      ;
0101          541  PTR_TXD_CLOCK11:
0101 200003 F  542      jb   PTR_XFER_TIMEDOUT, PTR_TXD_CLOCK_EXIT
0104 3000FA F  543      jnb  CTI_PTR_EXCLK, PTR_TXD_CLOCK11 ; Check for
clock LO
                544      ;
0107          545  PTR_TXD_CLOCK_EXIT:
0107 C2A9      546      clr  XFER_TIMEOUT_INTERRUPT
0109 C28C      547      clr  START_CTI_TIMEOUT
010B 22        548      ret
                549
                550      END
                ;
                ; End of CTI_XFER.A51
                ;

```

REGISTER BANK(S) USED: 0

ASSEMBLY COMPLETE, NO ERRORS FOUND

455

ACCESS_C1A67	ACTIM 1E39	ADMD 003D	ADMDBF	003E	AIR_TIME1D16
B1 1A74	BLOX1 1B01	BUF 1D8C			
BUF2 1D8D	B_CLK 1D8E	CALCNT 1D97	CALL_DAT1C34		
CALL_XFR1D2B	CC1 1ADD	CCC 1C50	CEL_NUM 1C6B		
CHANGE 1D16	CHECK_D 1BB7	CLK_RD DFF4	CLK_WR DFF0		
CLRF1 1AF5	CLR_AIR 1D55	COMM 1A7F	CTABLE 1BE3		
CURR_TIM1CAA	C_PTL 1D99	DATAR1C13	DATE 1D93	DEC13 1BA2	DEC14
1BA5 DIE	1C32 DONE_S	1BAC			
DONE_S1 1BAC	DONE_S2 1BAD	D_DATA 1C26	ERECBUF 0076		
ESOCNT 0069	EWDTBF 0366	EWDTRP 038E	EWDTWP		
038F					
EX_PTR 1BCD	FLAG1 002B	FLAG11 008A	FLAG15 008E	FLAG4	
1B50 FLAG7 0086	HOUR 1D91	INCO1 1A48			
INCOM 1A3F	INCOME 1A46	INC_CALL1A49	INC_UP1 1BC6	IND110	
BD79	INIT_CAL1B58	IN_USE1A20	IN_USE001A35		
KYSEND D34B	LB 1B57	LB1 1B62	LCOMM 1B6D	LNDONE	
1B5A LNIB 1B3C	LOAK 1B7E	LOCK 1E41			
LOCK1 1D03	LOCKA 1C80	LOCKB 1C95	LPNOV 1D59	LS_ONE 1B6B	
MBCDWG 1D03	MBC_SOFT1D03	MEM_FULL1D7C			
MIN 1D90	MONTH 1D94	ND1 1AA9	ND7 1AA2	NDONE1A6C	NIB1
1B4B NOKIDT	0080	NOV_SOFT1CEE			
NUM_BYTE1D55	NUM_CALL1D2B	OAK 1A90	POWER_DO1C31		
PROGRAM 1D55	PROG_PH 1CD3	PTR 1D40	PWR_DWN 1C31		
RD_NUM 1B1F	RD_NXT 1B23	READ_CLK1AA0	READ_CTP1B1F		
RESTART 1D11	RES_CALL1C55	RES_PTR 1C5E	RET_USE 1A63		
RET_USE11A3B	RET_USE21A5E	RTS1 1B1E	SB1 1AD6	SD_ONE	
1A7D SEC 1D8F	SER_NUM 1CD3	SETF0 1AEA			
SET_TBL 1B86	SET_TIME1CCB	SNDK05 B8A7	STINDO 0025		
STORE 1D60	STORE1 1B03	STORE_001D6D	STORE_EX1D7B		
STO_BY 1D98	STR_NUM 1D55	TEL_SOFT1CD9	TIME_TBL1A98		
UNLOCK 1D0A	USE_OFF 1A4E	VERSION 1D80			

E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

1      ;      Program Title:  Cellular Telephone RAM Based Controller
Firmware
2      ;
3      ;      Filename   :   PTR_RAMC.ASM
4      ;      Module Name : PTR_RAMC.OBJ
5      ;      Project #   :
6      ;      Author     :   Don Bloxson
7      ;      Date Created : September 9, 1990
8      ;      Reviewed By : Theodore W. Watler
9      ;      From       :   Parchment Designs
10     ;      For        :   Turner, Gold, France & Associates
11     ;      Date Reviewed: August 1, 8, 1991
12     ;      Version    :   A.00
13     ;
14     ;
15     ;      COPYRIGHT (C) 1991. ALL RIGHTS RESERVED
16     ;      Turner, Gold, France & Associates
17     ;
18     ;
19     ;
20     ;      PROGRAM FUNCTION
21     ;
22     ;
23     ;      PROGRAM DESCRIPTION
24     ;
25     ;
26     ;      REFERENCES
27     ;
28     ;      1. Mitsubishi Semiconductors SERIES 740 Software User's Manual
29     ;      2. Mitsubishi Semiconductors M37450M2-XXXSP/FP User's Manual
30     ;      3. Miscellaneous development documents, TGF, NOVATEL, TELEMAC
31     ;      & Don Bloxson
32     ;
33     ;      *****      MODULE HISTORY      *****
34     ;
35     ;      DPB 12-16-1990:
36     ;      Unkown modifications by Don Bloxson
37     ;
38     ;      TWW 08-08-1991
39     ;      Organization & Partition of this module for clarity
40     ;      and obvious firmware problems identification
41     ;
42     ;
43     ;      ;#####
```



```

E SEQ. LOC. OBJ..  ....*....1....*....2....*....SOURCE STATEMENT....5....*.

44          .PAGE
45          ;
46          ;          ROM CHANGES
47          ;
48          ;  USE CHIP RAM AREA F0 & F1 FOR TEMP PTR LOCATION TO USE
INDIRECT STORE
49          ;  8090 & 8091 TO NOP TO DISABLE CHECK SUM (FIX LATER)
50          ;  AB22 TO JSR FF48, TO JMP 1400, X_RCV, RECEIVED BYTE FROM MBC
THIS ALSO TAKES
51          ;  OUT AN INSTRUCTION THAT CAUSES THE PHONE TO ACT ON
THE BYTE RECEIVED
52          ;  AB43 TO JMP "FFC7" TEST IF EXTERNAL SEND IN PROCESS, BYPASS
"00" COMPARE
53          ;  SEE ROMMOD.ASM FOR JMP FFC7
54          ;  AB49 TO LDA #F00,X FROM LDA EWDTBF,X SO THAT IT POINTS TO
THE PHONE BUFFER
55          ;  NOW JSR 1360 TO ALLOW OP CODE MOD FOR ADDRESS
CHANGES
56          ;  AB84 TO JSR 1320 TO SEND NEXT BYTE IF NEEDED
57          ;  BA32 TO NOP NOP "ALLOWS ONLY 1-9 PHONE STORAGE"
58          ;  B8A4 TO JMP 1200 "TEST FOR IN COMING CALLS"
59          ;  BD66 TO JMP IN_USE (ADR 1000)
60          ;  BD6A TO JMP USE_OFF (ADR FFB2 TO 1050)
61          ;  FFFE TO 'INIT_CAL'
62          ;
63          ;
#####
64

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

65          .PAGE
66          ;
67          ; MEMORY CONSTANTS EQUATES
68          ;
69 003E      ADMDBF = 3EH          ;PTR BUFFER
70 0080      NOKIDT = 80H         ;PTR AUDIO SETTINGS
71 003D      ADMD  = 03DH         ;PTR AUDIO FLAGS
72 1E39      ACTIM = 1E39H        ;ACCUM TIMER LOCATION, ONE
BYTE
73 1E41      LOCK  = 1E41H        ;LOCK FLAG AREA
74 0076      ERECBUF = 76H        ;EXT DATA RECEIVE BYTE
LOCATION
75 0069      ESOCNT = 69H         ;EXT SEND PROCESS CNTR
76 038E      EWDTRP = 38EH        ;EXT READ POINTER ?
77 038F      EWDTWP = 38FH        ;EXT WRITE POINTER
78 0366      EWDTBF = 366H        ;EXT SEND BUFFER
79 D34B      KYSEND = 0D34BH      ;SEND KEY SUB
80 B8A7      SNDK05 = 0B8A7H      ;CALLED AFTER SNDKY
81 DFF4      CLK_RD = 0DFF4H      ;CLOCK READ ADDRESS
82 DFF0      CLK_WR = 0DFF0H      ;CLOCK WRITE ADDRESS
83 008A      FLAG11 = 8AH         ;ZERO PAGE FLAG, BIT 0 SET MEANS
INCOMING CALL
84 0086      FLAG7 = 86H         ;NOVATEL FLAGS
85 BD79      IND110 = 0BD79H      ;RETURN ADDRESS FOR IN_USE SUB
86 0025      STINDO = 25H         ;FLAG FOR IN_USE ILLUMINATION
87 002B      FLAG1 = 2BH         ;TELEMAC FLAGS, BIT 0 RESERVED
FOR NOVATEL
88 008E      FLAG15 = 8EH        ;TELEMAC FLAGS 0-7
89 1D80      VERSION = 1D80H     ;LOCATION OF SOFTWARE VERSION
NUMBER
90 1D8C      BUF  = 1D8CH        ;WORKING AREA,USED TO STORE
BYTE IN ACCESS_CLK
91 1D8D      BUF2 = 1D8DH        ;DITTO
92 1D8E      B_CLK = 1D8EH       ;8 BYTE STORAGE BUFFER FOR
CLK/CAL DATA
93          ;
94 1D8F      SEC  = 1D8FH        ;TEMP BUF FOR SECONDS
95 1D90      MIN  = 1D90H        ;TEMP BUF FOR MINUTES
96 1D91      HOUR = 1D91H        ;TEMP BUF FOR HOUR
97          ;1D92H
98          ;
99 1D93      DATE = 1D93H        ;TEMP BUF FOR DATE
100 1D94      MONTH = 1D94H      ;TEMP BUF FOR MONTH
101          ;1D95H
102          ;
103          ;1D96H              ;HIGH BYTE OF CALL COUNTER
104 1D97      CALCNT = 1D97H      ;NUMBER OF PHONE CALLS SINCE
DOWNLOADING
105          ;
106          ;1D98H              ;OP CODE STA "8D" FOR SUB ROUTINE STORE
107 1D99      C_PTL = 1D99H      ;LOCATION OF PHONE POINTER, LOW
BYTE
108          ;1D9AH              ;" " " " , HIGH BYTE
109          ;1D9BH              ;OP CODE RTS "60" FOR SUB ROUTINE STORE

```


110 1BCD EX_PTR = 1BCDH
ADDRESS BYTE FOR OP CODE MOD
111

;* = CHANGES UPPER BYTE OF EX

E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

112		.PMOD	
113		;	
114		;	*= \$0EF0
115			LOAD "TELEMAC"
116		*= \$0F90	;STORE #1 RECALL NUMBER
117	0F90 EF	.BYTE \$EF	;FLAG FOR NO PHONE CALLS
118			

E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

119      .PAGE
120      ;
121      *= $1A20
122      ; *= $1000
123      ;
124      ;*<
125      ; NAME: In Use
126      ;
127      ; DESCRIPTION:
128      ;         Function to process in use illumination
129      ;
130      ; CALL:
131      ; ARGUMENTS:
132      ; MODIFIES:
133      ; RETURNS:
134      ; HISTORY:
135      ;*>
136      ;
137      IN_USE:
138      ;
139 1A20 12      CLT                      ;WARNING, DOES NOT RESTORE
140 1A21 472517  BBS 2,STINDO,RET_USE1    ;PHONE UPDATES IN USE OFTEN
141 1A24 C78E14  BBS 6,FLAG15,RET_USE1    ;MEMORY FULL
142 1A27 78      SEI                      ;DISABLE INTR
143 1A28 4F25     SEB 2,STINDO             ;IN USE ON
144 1A2A 20671A  JSR ACCESS_CLK           ;READ AND STORE TIME/CAL
145 1A2D 20A01A  JSR READ_CLK
146      ;
147      ;<<<TWW Aug-11-1991      Move this section of code from the end of the call
(USE_OFF)
148      ;                               As per GM request in case the battery is removed before
149      ;                               ending the call, to account for call made count.
150      ;
151 1A30 EE971D      inc      CALCNT                      ; TWW:Increment the call
counter
152 1A33 F014      beq      INC_CALLS_MSB                ; TWW:If rollover carry
to MSB
153      ;
154      IN_USE00:                                       ; TWW:Return here after count
rollover
155      ;
156      ;>>>TWW
157      ;
158 1A35 472B03     BBS 2,FLAG1,RET_USE1    ;INCOMING CALL
159 1A38 201F1B     JSR RD_NUM
160      ;
161      RET_USE1:
162 1A3B 4F25     SEB 2,STINDO             ;IN USE ON
163 1A3D 8024     BRA RET_USE
164      ;
165      INCOM:
166 1A3F 078A04     BBS 0,FLAG11,INCOME     ;*****COMES HERE WHEN THE SEND
KEY IS PRESSED
167 1A42 5F2B     CLB 2,FLAG1
168 1A44 8002     BRA INCO1

```



```
169      ;  
170      INCOME:  
171 1A46 4F2B      SEB 2,FLAG1      ;SET INCOMING FLAG  
172      ;
```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

173	INCO1:
174 1A48 60	RTS
175	*****
176	

E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```
177 .PAGE
178 ;
179 ;*<
180 ; NAME:
181 ;
182 ; DESCRIPTION:
183 ; CALL:
184 ; ARGUMENTS:
185 ; MODIFIES:
186 ; RETURNS:
187 ; HISTORY:
188 ;*>
189 ;
190 ;
191 ;<<<TWW Aug-11-1991
192 ;
193 INC_CALLS_MSB:
194 1A49 EE961D inc CALCNT-1 ; TWW:Increment call
count MSB
195 1A4C 80E7 bra IN_USE00 ; TWW:Return to
complete function
196 ;
197 ;>>>TWW
198
```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

199          .PAGE
200          ;   *= $1050
201          ;
202          ;*<
203          ;   NAME: Use Off
204          ;
205          ;   DESCRIPTION:
206          ;   CALL:
207          ;   ARGUMENTS:
208          ;   MODIFIES:
209          ;   RETURNS:
210          ;   HISTORY:
211          ;*>
212          ;
213          USE_OFF:
214 1A4E 12      CLT                      ;WARNING, DOES NOT RESTORE
215 1A4F 57250C  BBC 2,STINDO,RET_USE2
216 1A52 C78E09  BBS 6,FLAG15,RET_USE2      ;MEMORY FULL DO NOT STORE
217 1A55 78      SEI                      ;DISABLE INTR
218 1A56 5F25     CLB 2,STINDO              ;IN USE OFF
219 1A58 20671A   JSR ACCESS_CLK            ;READ AND STORE TIME/CAL
220 1A5B 20A01A   JSR READ_CLK
221          ;
222          ;<<<TWW Aug-11-1991      Move this section of code to the beginning of the
call
223          ;                      As per GM request in case the battery is removed before
224          ;                      ending the call.
225          ;
226          ;   INC CALCNT              ;INCREMENT CALL COUNTER
227          ;   BEQ CALCNT1             ;MORE THAN 255?
228          ;
229          ;>>>TWW
230          ;
231          RET_USE2:
232 1A5E 203F1A   JSR INCOM                ;FLAG FOR INCOMING CALLS
233 1A61 5F25     CLB 2,STINDO              ;IN USE OFF
234          ;
235          RET_USE:
236 1A63 58      CLI                      ;ENABLE INTR
237 1A64 4C79BD   JMP IND110
238          ;
239          ;<<<TWW Aug-11-1991      Moved to end of IN_USE function above
240          ;
241          ;CALCNT1:
242          ;   INC CALCNT-1             ;UPPER
243          ;   BRA RET_USE2
244          ;
245          ;>>>TWW
246

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

247          .PAGE
248          ;
249          ;*<
250          ; NAME:
251          ;
252          ; DESCRIPTION:
253          ; CALL:
254          ; ARGUMENTS:
255          ; MODIFIES:
256          ; RETURNS:
257          ; HISTORY:
258          ;*>
259          ;
260          ACCESS_CLK:
261          ;
262 1A67 ADF4DF  LDA CLK_RD          ;START SEQ BY A2 HIGH
263 1A6A A200   LDX #0              ;TABLE POINTER
264          ;
265          NDONE:
266 1A6C BD981A  LDA TIME_TBL,X
267 1A6F 8D8C1D  STA BUF
268 1A72 A008   LDY #8              ;BIT CNTR
269          ;
270          B1:
271 1A74 8E8D1D  STX BUF2           ;SAVE X
272 1A77 0304   BBS 0,A,SD_ONE      ;SEND ONE?
273 1A79 A200   LDX #0              ;ACCESS CLOCK CHIP
274 1A7B 8002   BRA COMM
275          ;
276          SD_ONE:
277 1A7D A201   LDX #1
278          ;
279          COMM:
280 1A7F BDF0DF  LDA CLK_WR,X
281 1A82 AE8D1D  LDX BUF2           ;RESTORE X, TABLE POINTER
282 1A85 88     DEY
283 1A86 C000   CPY #0              ;DONE WITH THIS BYTE?
284 1A88 D006   BNE OAK
285 1A8A E8     INC                 ;INC TABLE POINTER
286 1A8B E008   CPX #8              ;DONE?
287 1A8D D0DD   BNE NDONE
288          ;
289 1A8F 60     RTS
290          ;
291          OAK:
292 1A90 6E8C1D  ROR BUF             ;ROTATE RIGHT ONE BIT

293 1A93 AD8C1D  LDA BUF
294 1A96 80DC   BRA B1
295

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```
296          .PAGE
297          ;
298          ;*<
299          ;  NAME:
300          ;
301          ;  DESCRIPTION:
302          ;  CALL:
303          ;  ARGUMENTS:
304          ;  MODIFIES:
305          ;  RETURNS:
306          ;  HISTORY:
307          ;*>
308          ;
309          TIME_TBL:
310 1A98 C5      .BYTE 0C5H          ;TABLE TO ACCESS TIME
311 1A99 3A      .BYTE 3AH
312 1A9A A3      .BYTE 0A3H
313 1A9B 5C      .BYTE 5CH
314 1A9C C5      .BYTE 0C5H
315 1A9D 3A      .BYTE 3AH
316 1A9E A3      .BYTE 0A3H
317 1A9F 5C      .BYTE 5CH
318          ;
319
```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

320          .PAGE
321          ;
322          ;*<
323          ; NAME:
324          ;
325          ; DESCRIPTION:
326          ; CALL:
327          ; ARGUMENTS:
328          ; MODIFIES:
329          ; RETURNS:
330          ; HISTORY:
331          ;*>
332          ;
333          READ_CLK:
334 1AA0 A000      LDY #0          ;BYTE CNTR
335          ;
336          ND7:
337 1AA2 A208      LDX #8          ;BIT POINTER
338 1AA4 A900      LDA #0          ;CLR BUFFER
339 1AA6 8D8C1D    STA BUF
340          ;
341          ND1:
342 1AA9 18        CLC              ;CLR CARRY FLAG
343 1AAA 6E8C1D    ROR BUF          ;ROTATE FOR NEXT BIT
344 1AAD ADF4DF      LDA CLK_RD      ;READ D0
345 1AB0 2901      AND #00000001B    ;ISOLATE D0
346 1AB2 6A        ROR A            ;GET LSB TO MSBIT
347 1AB3 6A        ROR A
348 1AB4 0D8C1D    ORA BUF
349 1AB7 8D8C1D    STA BUF          ;STORE
350 1ABA CA        DEX
351 1ABB D0EC      BNE ND1
352          ;
353 1ABD AD8C1D      LDA BUF
354 1AC0 998E1D    STA B_CLK,Y      ;STORE TIME DATA (TEMP)
355 1AC3 C8        INY              ;BYTE CNTR
356 1AC4 C008      CPY #8            ;EQUAL TO 8 BYTES
357 1AC6 D0DA      BNE ND7          ;DONE WITH READ?
358          ;
359 1AC8 AD941D    LDA MONTH        ;SQUEEZE CLK/CAL DATA TO 5 BYTES
360 1ACB 8309      BBS 4,A,SB1      ;SET BIT FOR 10 MONTHS
361 1ACD AD931D    LDA DATE
362 1AD0 DB        CLB 6,A
363 1AD1 8D931D    STA DATE
364 1AD4 8007      BRA CC1
365          ;
366          SB1:
367 1AD6 AD931D    LDA DATE
368 1AD9 CB        SEB 6,A
369 1ADA 8D931D    STA DATE
370          ;

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

371          PAGE
372          ;
373          CC1:
374 1ADD 872B3E  BBS 4,FLAG1,RTS1  ;EX REQUEST, DO NOT STORE TIME
375 1AE0 772507  BBC 3,STINDO,SETF0 ;SET ROAM BIT ?, FLAG 0
376 1AE3 AD931D  LDA DATE
377 1AE6 EB      SEB 7,A           ;SET ROAM BIT
378 1AE7 8D931D  STA DATE
379          ;
380          SETF0:
381 1AEA 643E     TST ADMDBF        ;BEEP ON?
382 1AEC F007     BEQ CLRF1
383 1AEE AD911D   LDA HOUR
384 1AF1 EB      SEB 7,A
385 1AF2 8D911D   STA HOUR
386          ;
387          CLRF1:
388 1AF5 AD941D   LDA MONTH        ;SET UPPER NIBBLE TO MONTH FOR INUSE
389 1AF8 472506   BBS 2,STINDO,BLOX1 ;FIX DO NOT TEST THIS BIT
390 1AFB 290F     AND #00001111B   ;CLR BIT 4
391 1AFD 09E0     ORA #11100000B   ;INUSE OFF SET TO "E"
392 1AFF 8002     BRA STORE1
393          ;
394          BLOX1:
395 1B01 09F0     ORA #11110000B   ;INUSE ON SET TO "F"
396          ;
397          STORE1:
398 1B03 20601D   JSR STORE        ;STORE DATA, STORE SQUEEZED CLK/CAL
TO RAM
399 1B06 AD931D   LDA DATE        ;INC PTR
400 1B09 20601D   JSR STORE
401 1B0C AD911D   LDA HOUR
402 1B0F 20601D   JSR STORE
403 1B12 AD901D   LDA MIN
404 1B15 20601D   JSR STORE
405 1B18 AD8F1D   LDA SEC
406 1B1B 20601D   JSR STORE
407          ;
408          RTS1:
409 1B1E 60       RTS
410          ;
411

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

412          .PAGE
413          ;
414          ;*<
415          ; NAME:
416          ;
417          ; DESCRIPTION:
418          ; CALL:
419          ; ARGUMENTS:
420          ; MODIFIES:
421          ; RETURNS:
422          ; HISTORY:
423          ;*>
424          ;
425          READ_CTPHONE_NUMBER:
426          RD_NUM:
427 1B1F A208      LDX #8          ;OFFSET FROM 394
428                      ;ODD/EVEN NIBBLE
429 1B21 3F2B      CLB 1,FLAG1    ;PHONE UPDATES "IN USE" OFTEN
430          ;
431          RD_NXT:
432 1B23 BD9403     LDA 394H,X      ;READ NUMBER
433 1B26 29F0      AND #0F0H      ;CHECK UPPER NIBBLE "4"
434 1B28 C940      CMP #40H
435 1B2A D024      BNE FLAG4       ;ERROR "4" NOT IN UPPER NIBBLE
436          ;
437 1B2C BD9403     LDA 394H,X      ;GET NUMBER
438 1B2F 290F      AND #0FH       ;STRIP UPPER NIBBLE (4)
439 1B31 272B08     BBS 1,FLAG1,LNIB ;LOWER NIBBLE?
440 1B34 18        CLC           ;CLEAR CARRY FLAG BEFORE ROTATE
441 1B35 2A        ROL A         ;MOVE LOWER NIBBLE TO UPPER NIBLLE
442 1B36 2A        ROL A
443 1B37 2A        ROL A
444 1B38 2A        ROL A
445 1B39 48        PHA           ;SAVE ACC
446 1B3A 800F      BRA NIB1
447          ;
448          LNIB:
449 1B3C 8D8C1D     STA BUF        ;ADD LOWER NIBBLE
450 1B3F 68        PLA
451 1B40 0D8C1D     ORA BUF
452 1B43 20601D     JSR STORE
453 1B46 3F2B      CLB 1,FLAG1
454 1B48 E8        INX
455 1B49 80D8      BRA RD_NXT
456          ;
457          NIB1:
458 1B4B E8        INX
459 1B4C 2F2B      SEB 1,FLAG1    ;SET LOWER NIBBLE FLAG
460 1B4E 80D3      BRA RD_NXT
461

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```
462          .PAGE
463          ;*****
464          FLAG4:
465 1B50 372B04  BBC 1,FLAG1,LB          ;LOWER NIBBLE?
466 1B53 68     PLA
467 1B54 20601D JSR STORE
468          ;
469          LB:
470 1B57 60     RTS
471
```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

472          .PAGE
473          ;
474          ;*<
475          ;  NAME:
476          ;
477          ;  DESCRIPTION:
478          ;  CALL:
479          ;  ARGUMENTS:
480          ;  MODIFIES:
481          ;  RETURNS:
482          ;  HISTORY:
483          ;*>
484          ;
485          INIT_CAL:          ;LOAD CLK/CAL
486 1B58 A200      LDX #0      ;TABLE POINTER
487          ;
488          LNDONE:
489 1B5A BD861B    LDA SET_TBL,X
490 1B5D 8D8C1D    STA BUF
491 1B60 A008      LDY #8      ;BIT CNTR
492          ;
493 1B62 8E8D1D    LB1:      STX BUF2          ;SAVE X
494 1B65 0304      BBS 0,A,LS_ONE          ;SEND ONE ?
495 1B67 A200      LDX #0
496 1B69 8002      BRA LCOMM
497          ;
498          LS_ONE:
499 1B6B A201      LDX #1
500          ;
501          LCOMM:
502 1B6D BDF0DF    LDA CLK_WR,X
503 1B70 AE8D1D    LDX BUF2          ;RESTORE X, TABLE POINTER
504 1B73 88        DEY
505 1B74 C000      CPY #0          ;DONE WITH THIS BYTE?
506 1B76 D006      BNE LOAK
507 1B78 E8        INX          ;INC TABLE POINTER
508 1B79 E008      CPX #8          ;DONE?
509 1B7B D0DD      BNE LNDONE
510          ;
511 1B7D 60        RTS          ;TEST ONLY, ONE SUB IN REAL VERSION
512          ;
513          LOAK:
514 1B7E 6E8C1D    ROR BUF          ;ROTATE RIGHT ONE BIT

515 1B81 AD8C1D    LDA BUF
516 1B84 80DC      BRA LB1
517

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

518          PAGE
519          ;
520          ;*<
521          ;  NAME:
522          ;
523          ;  DESCRIPTION:
524          ;  CALL:
525          ;  ARGUMENTS:
526          ;  MODIFIES:
527          ;  RETURNS:
528          ;  HISTORY:
529          ;*>
530          ;
531          ;
532          SET_TBL:
533 1B86 00      .BYTE 00000000B      ;SET TIME, SEE PG 64 OF THE
534 1B87 00      .BYTE 00000000B      ;1987 DATA BOOK, DALLAS SEMI
535 1B88 45      .BYTE 01000101B      ;MINUTES
536 1B89 A5      .BYTE 10100101B      ;12/24 SELECT
537 1B8A 17      .BYTE 00010111B      ;RESET/DAY
538 1B8B 22      .BYTE 00100010B      ;DATE
539 1B8C 09      .BYTE 00001001B      ;MONTH
540 1B8D 90      .BYTE 10010000B      ;YEAR
541

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

542          .PAGE
543          ;
544          ;*<
545          ; NAME:
546          ;
547          ; DESCRIPTION:
548          ; CALL:
549          ; ARGUMENTS:
550          ; MODIFIES:
551          ; RETURNS:
552          ; HISTORY:
553          ;*>
554          ;
555          ;*****
556          ;  *= 1200H                ;FLAG TELLS IF CALL WAS INCOMING
557          ;TEMP, COMES HERE WHEN SEND KEY IS PRESSED
558 1B8E 204BD3 JSR KYSEND                ;RESTORE, FROM "B8A4"
559 1B91 4CA7B8 JMP SNDK05                ;RESTORE
560
561          ;***** EXTERNAL SEND ROUTINE HERE, REAL! *****
562          ;***** ADDR IN EWDTRP + BUF FROM EX_PTR & LSB END OF ADDR IN
BUF2 ***
563          ;  *= 1320H
564 1B94 3C0069 LDM #0,ESOCNT                ;RESTORE ROM PATCH
565 1B97 772B13 BBC 3,FLAG1,DONE_S2          ;INIT BY PTR800 OR TELEMAT
566 1B9A AD8C1D LDA BUF                      ;NEXT BYTE TO SEND
567 1B9D CD8D1D CMP BUF2
568 1BA0 F015   BEQ CHECK_D                ;DONE SENDING?
569          ;
570          DEC13:
571 1BA2 3A     INC A
572 1BA3 F021   BEQ INC_UP1
573          ;
574          DEC14:
575 1BA5 8D8C1D STA BUF
576 1BA8 8D8E03 STA EWDTRP
577 1BAB 60     RTS
578          ;
579          DONE_S:
580          DONE_S1:
581 1BAC 68     PLA                        ;RESTORE STACK FROM CHECL_D ROUTINE
582          ;
583          DONE_S2:
584 1BAD A900   LDA #$00                  ;CLEAR EXT WRITE POINTER (NOVATEL)
585 1BAF 8D8E03 STA EWDTRP
586 1BB2 7F2B   CLB 3,FLAG1              ;DONE TELEMAT EXT SEND
587 1BB4 9F8E   CLB 4,FLAG15
588 1BB6 60     RTS
589

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

590          .PAGE
591          ;
592          ;*<
593          ; NAME:
594          ;
595          ; DESCRIPTION:
596          ; CALL:
597          ; ARGUMENTS:
598          ; MODIFIES:
599          ; RETURNS:
600          ; HISTORY:
601          ;*>
602          ;
603          CHECK_D:
604 1BB7 48      PHA          ;SAVE ACC
605 1BB8 978EF1  BBC 4,FLAG15,DONE_S      ;NOT PHONE NUMBER DOWN LOAD
606 1BBB ADCD1B      LDA EX_PTR
607 1BBE CD9A1D  CMP C_PTL+1      ;UPPER BYTE OF PTR = ALSO?
608 1BC1 F0E9      BEQ DONE_S
609 1BC3 68      PLA
610 1BC4 80DC      BRA DEC13
611          ;
612          INC_UP1:
613 1BC6 EEC1B  INC EX_PTR      ;INC UPPER BYTE OF DATA POINTER
614 1BC9 80DA      BRA DEC14
615
616          ;  *= 1360H
617          ;***** LOCATION TO POINT TO BEGINNING OF BUFFER FOR
EXTERNAL OUT *****
618 1BCB BD000F  LDA $0F00H,X
619 1BCE 60      RTS
620

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

621          .PAGE
622          ;
623          ;*<
624          ; NAME:
625          ;
626          ; DESCRIPTION:
627          ; CALL:
628          ; ARGUMENTS:
629          ; MODIFIES:
630          ; RETURNS:
631          ; HISTORY:
632          ;*>
633          ;
634          ; *= 1400H
635
636 1BCF A78E41  BBS 5,FLAG15,DATAR ;SET CLOCK DATA BYTE?
637 1BD2 0A     ASL A                ;MULTIPLY BY TWO
638 1BD3 AA     TAX                  ;MOVE A TO X
639 1BD4 BDE31B LDA CTABLE,X          ;ASSUME COMMAND IN X REG
640 1BD7 8D8C1D STA BUF
641 1BDA BDE41B LDA CTABLE+1,X
642 1BDD 8D8D1D STA BUF+1
643 1BE0 6C8C1D JMP (BUF)
644

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

645          .PAGE
646          ;
647          ;*****
648          ;          COMMAND TABLE (FROM MBC)
649          ;*****
650 1BE3 6B1C  CTABLE: .WORD CEL_NUM          ;0, PHONE'S PHONE NUMBER
651 1BE5 341C  .WORD CALL_DATA          ;1, SEND CALL DATA
652 1BE7 AA1C  .WORD CURR_TIME          ;2, CURRENT TIME AND DATE
653 1BE9 CB1C  .WORD SET_TIME          ;3, SET TIME AND DATE
654 1BEB D31C  .WORD PROG_PH          ;4, REPROGGRAM PHONE
655 1BED 551D  .WORD NUM_BYTES          ;5, TURN MBC CRADLE OFF (MBC)
656 1BEF D91C  .WORD TEL_SOFT          ;6, TELEMAT SOFTWARE VERSION
657 1BF1 EE1C  .WORD NOV_SOFT          ;7, PTR800 SOFTWARE VERSION
658 1BF3 031D  .WORD MBC_SOFT          ;8, MBC SOFTWARE VERSION, MBC
TO RESPOND
659 1BF5 031D  .WORD MBCDWG          ;9, MBC HARDWARE VERSION, MBC
TO RESPOND
660 1BF7 031D  .WORD LOCK1          ;A, LOCK THE PHONE
661 1BF9 0A1D  .WORD UNLOCK          ;B, UNLOCK THE PHONE
662 1BFB 161D  .WORD CHANGE          ;C, TBD
663 1BFD 161D  .WORD AIR_TIME          ;D, SEND CUMULATIVE AIR TIME
COUNTER
664 1BFF 311C  .WORD PWR_DWN          ;E, POWER DOWN PHONE, TBD
665 1C01 2B1D  .WORD NUM_CALLS          ;F, PHONE CALL COUNTER
666 1C03 401D  .WORD PTR          ;10, SEND POINTER
667 1C05 551D  .WORD PROGRAM          ;11, PROGRAM PHONE, LOCATE IN
ROM
668 1C07 551D  .WORD STR_NUM          ;12, MBC ONLY
669 1C09 5E1C  .WORD RES_PTR          ;13, RESET MEMORY PTR (SEE #10)
670 1C0B 551C  .WORD RES_CALL          ;14, RESET CALL CNTR
671 1C0D 551D  .WORD CLR_AIR          ;15, RESET CUMULATIVE AIR TIMER
672 1C0F 801C  .WORD LOCKA          ;16, SEND LOCK A CODE
673 1C11 951C  .WORD LOCKB          ;17, SEND LOCK B CODE
674

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

675          .PAGE
676          ;
677          ;*<
678          ; NAME:
679          ;
680          ; DESCRIPTION:
681          ; CALL:
682          ; ARGUMENTS:
683          ; MODIFIES:
684          ; RETURNS:
685          ; HISTORY:
686          ;*>
687          ;
688          DATAR:
689 1C13 AE8C1D  LDX BUF          ;STORE DATA
690 1C16 C9FF    CMP #0FFH      ;TIMEOUT ERROR SENT BY THE MBC
691 1C18 F017    BEQ PWR_DWN
692 1C1A 9D861B  STA SET_TBL,X
693 1C1D E8      INX
694 1C1E E008    CPX #8         ;DONE?
695 1C20 F004    BEQ D_DATA
696 1C22 8E8C1D  STX BUF
697 1C25 60      RTS
698          ;
699          D_DATA:
700 1C26 BF8E    CLB 5,FLAG15   ;YES
701 1C28 78      SEI           ;DISABLE INTR
702 1C29 20671A  JSR ACCESS_CLK ;ACCESS CLOCK CHIP
703 1C2C 20581B  JSR INIT_CAL  ;TEMP
704 1C2F 58      CLI           ;ENABLE INTERRUPTS
705 1C30 60      RTS
706

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```
707          .PAGE
708          ;
709          ;*<
710          ; NAME:
711          ;
712          ; DESCRIPTION:
713          ; CALL:
714          ; ARGUMENTS:
715          ; MODIFIES:
716          ; RETURNS:
717          ; HISTORY:
718          ;*>
719          ;
720          POWER_DOWN:
721          PWR_DWN:
722 1C31 78      SEI
723          ;
724          DIE:
725 1C32 80FE    BRA DIE                      ; Infinite LOOP, Interrupts Disabled
726
```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

727          .PAGE
728          ;
729          ;*<
730          ; NAME:
731          ;
732          ; DESCRIPTION:
733          ; CALL:
734          ; ARGUMENTS:
735          ; MODIFIES:
736          ; RETURNS:
737          ; HISTORY:
738          ;*>
739          ;
740          CALL_DATA:
741 1C34 A90F    LDA #0FH
742 1C36 8DCD1B STA EX_PTR          ;SET UPPER BYTE OF POINTER
743 1C39 A990    LDA #90H          ;STARTING POINT OF PHONE MEMORY
744 1C3B 8F8E    SEB 4,FLAG15      ;TELEMAC SEND EXT FLAG
745 1C3D 6F2B    SEB 3,FLAG1
746 1C3F 8D8C1D STA BUF
747 1C42 8D8E03 STA EWDTRP
748
749 1C45 AD991D  LDA C_PTL          ;GET LOW BYTE OF PHONE POINTER
750 1C48 C990    CMP #90H
751 1C4A D004    BNE CCC
752 1C4C 8D8D1D  STA BUF2          ;STORE TO KNOW WHEN DONE
753 1C4F 60      RTS
754

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

755          .PAGE
756          ;
757          ;*<
758          ;  NAME:
759          ;
760          ;  DESCRIPTION:
761          ;  CALL:
762          ;  ARGUMENTS:
763          ;  MODIFIES:
764          ;  RETURNS:
765          ;  HISTORY:
766          ;*>
767          ;
768          CCC:
769 1C50 1A      DEC A
770 1C51 8D8D1D  STA BUF2
771 1C54 60      RTS
772          ;
773

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

774          .PAGE
775          ;
776          ;*<
777          ;  NAME:
778          ;
779          ;  DESCRIPTION:
780          ;  CALL:
781          ;  ARGUMENTS:
782          ;  MODIFIES:
783          ;  RETURNS:
784          ;  HISTORY:
785          ;*>
786          ;
787          RES_CALL:
788 1C55 A900    LDA #$00          ;RESET CALL CNTR
789 1C57 8D971D STA CALCNT
790 1C5A 8D961D STA CALCNT-1
791 1C5D 60     RTS
792          ;
793

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```
794          .PAGE
795          ;
796          ;*<
797          ;  NAME:
798          ;
799          ;  DESCRIPTION:
800          ;  CALL:
801          ;  ARGUMENTS:
802          ;  MODIFIES:
803          ;  RETURNS:
804          ;  HISTORY:
805          ;*>
806          ;
807          RES_PTR:
808 1C5E A990    LDA #$90
809 1C60 8D991D STA C_PTL
810 1C63 A90F    LDA #$0F          ;RESET POINTER
811 1C65 8D9A1D STA C_PTL+1
812 1C68 DF8E    CLB 6,FLAG15      ;CLEAR MEMORY FULL FLAG
813 1C6A 60      RTS
814
```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

815          PAGE
816          ;
817          ;*<
818          ; NAME:
819          ;
820          ; DESCRIPTION:
821          ; CALL:
822          ; ARGUMENTS:
823          ; MODIFIES:
824          ; RETURNS:
825          ; HISTORY:
826          ;*>
827          ;
828          ;
829          CEL_NUM:                ;PHONE'S PHONE NUMBER
830 1C6B A91F      LDA #1FH
831 1C6D 8DCD1B    STA EX_PTR        ;SET UPPER BYTE OF POINTER
832 1C70 A953      LDA #53H          ;STARTING ADDR OF CALL PTR (MUST IN 1F
HIGH BYTE PAGE)
833 1C72 6F2B      SEB 3,FLAG1      ;TELEMAC SEND EXT FLAG
834 1C74 8D8C1D    STA BUF
835 1C77 8D8E03    STA EWDTRP
836 1C7A A959      LDA #$59          ;ENDING ADDR OF CALL PTR
837 1C7C 8D8D1D    STA BUF2
838 1C7F 60        RTS
839          ;
840

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```
841          PAGE
842          ;
843          ;*<
844          ;  NAME:
845          ;
846          ;  DESCRIPTION:
847          ;  CALL:
848          ;  ARGUMENTS:
849          ;  MODIFIES:
850          ;  RETURNS:
851          ;  HISTORY:
852          ;*>
853          ;
854          LOCKA:
855 1C80 A91F    LDA #1FH
856 1C82 8DCD1B  STA EX_PTR          ;SET UPPER BYTE OF POINTER
857 1C85 A95F    LDA #5FH          ;STARTING ADDR OF CALL PTR (MUST IN 1F
HIGH BYTE PAGE)
858 1C87 6F2B    SEB 3,FLAG1        ;TELEMAC SEND EXT FLAG
859 1C89 8D8C1D  STA BUF
860 1C8C 8D8E03  STA EWDTRP
861 1C8F A961    LDA #$61          ;ENDING ADDR OF CALL PTR
862 1C91 8D8D1D  STA BUF2
863 1C94 60      RTS
864          ;
865
```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

866          .PAGE
867          ;
868          ;*<
869          ;  NAME:
870          ;
871          ;  DESCRIPTION:
872          ;  CALL:
873          ;  ARGUMENTS:
874          ;  MODIFIES:
875          ;  RETURNS:
876          ;  HISTORY:
877          ;*>
878          ;
879          LOCKB:
880 1C95 A91F    LDA #1FH
881 1C97 8DCD1B  STA EX_PTR      ;SET UPPER BYTE OF POINTER
882 1C9A A964    LDA #64H      ;STARTING ADDR OF CALL PTR (MUST IN 1F
HIGH BYTE PAGE)
883 1C9C 6F2B    SEB 3,FLAG1    ;TELEMAC SEND EXT FLAG
884 1C9E 8D8C1D  STA BUF
885 1CA1 8D8E03  STA EWDTRP
886 1CA4 A966    LDA #$66      ;ENDING ADDR OF CALL PTR
887 1CA6 8D8D1D  STA BUF2
888 1CA9 60      RTS
889          ;
890

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

891          PAGE
892          ;
893          ;*<
894          ;  NAME:
895          ;
896          ;  DESCRIPTION:
897          ;  CALL:
898          ;  ARGUMENTS:
899          ;  MODIFIES:
900          ;  RETURNS:
901          ;  HISTORY:
902          ;*>
903          ;
904          CURR_TIME:                ;CURRENT TIME AND DATE
905 1CAA 78      SEI                  ;DISABLE INTR
906 1CAB 8F2B    SEB 4,FLAG1         ;DO NOT STORE TIME IN CALL MEMORY
907 1CAD 20671A  JSR ACCESS_CLK      ;ACCESS CLOCK CHIP
908 1CB0 20A01A  JSR READ_CLK        ;READ CLOCK, STORE DATA IN
909 1CB3 9F2B    CLB 4,FLAG1
910 1CB5 A91D    LDA #1DH
911 1CB7 8DCD1B  STA EX_PTR          ;SET UPPER BYTE OF POINTER
912 1CBA A98E    LDA #8EH           ;STARTING ADDR
913 1CBC 6F2B    SEB 3,FLAG1        ;TELEMAC SEND EXT FLAG
914 1CBE 8D8C1D  STA BUF
915 1CC1 8D8E03  STA EWDTRP
916 1CC4 A996    LDA #$8E+8         ;ENDING ADDR
917 1CC6 8D8D1D  STA BUF2
918 1CC9 58      CLI                ;ENABLE INTERRUPTS
919 1CCA 60      RTS
920          ;
921

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```
922          .PAGE
923          ;
924          ;*<
925          ; NAME:
926          ;
927          ; DESCRIPTION:
928          ; CALL:
929          ; ARGUMENTS:
930          ; MODIFIES:
931          ; RETURNS:
932          ; HISTORY:
933          ;*>
934          ;
935          SET_TIME:                                ;SET TIME AND DATE
936 1CCB AF8E      SEB 5,FLAG15
937 1CCD A200      LDX #0                            ;STORE INIT DATA AREA
938 1CCF 8E8C1D    STX BUF
939 1CD2 60        RTS
940          ;
941
```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

942		PAGE	
943		;	
944		;*<	
945		; NAME:	
946		;	
947		; DESCRIPTION:	
948		; CALL:	
949		; ARGUMENTS:	
950		; MODIFIES:	
951		; RETURNS:	
952		; HISTORY:	
953		;*>	
954		;	
955		SER_NUM: ;RTS	PHONE'S SERIAL NUMBER
956		PROG_PH:	;ALLOWS REPROGGRAMMING OF PHONE
957	1CD3 A900	LDA #00H	;PUT 0 WHERE "V" WAS
958	1CD5 8D801D	STA \$1D80	
959	1CD8 60	RTS	
960		;	
961			

E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

962          .PAGE
963          ;
964          ;*<
965          ; NAME:
966          ;
967          ; DESCRIPTION:
968          ; CALL:
969          ; ARGUMENTS:
970          ; MODIFIES:
971          ; RETURNS:
972          ; HISTORY:
973          ;*>
974          ;
975          TEL_SOFT:                                ;TELEMAC SOFTWARE VERSION
976 1CD9 A91D      LDA #1DH
977 1CDB 8DCD1B      STA EX_PTR                        ;SET UPPER BYTE OF POINTER
978 1CDE A980      LDA #80H                            ;STARTING ADDR OF VERSION NUMBER
(MUST IN 0F HIGH BYTE PAGE)
979 1CE0 6F2B      SEB 3,FLAG1                        ;TELEMAC SEND EXT FLAG
980 1CE2 8D8C1D      STA BUF
981 1CE5 8D8E03      STA EWDTRP
982 1CE8 A98C      LDA #8CH                            ;ENDING ADDR OF VERSION NUMBER
983 1CEA 8D8D1D      STA BUF2
984 1CED 60         RTS
985          ;
986

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

987          PAGE
988          ;
989          ;*<
990          ; NAME:
991          ;
992          ; DESCRIPTION:
993          ; CALL:
994          ; ARGUMENTS:
995          ; MODIFIES:
996          ; RETURNS:
997          ; HISTORY:
998          ;*>
999          ;
1000         NOV_SOFT:                                ;PTR800 SOFTWARE VERSION
1001 1CEE A9FF    LDA #0FFH
1002 1CF0 8DCD1B  STA EX_PTR                        ;SET UPPER BYTE OF POINTER
1003 1CF3 A9D0    LDA #0D0H                        ;STARTING ADDR OF VERSION NUMBER
(MUST IN 0F HIGH BYTE PAGE)
1004 1CF5 6F2B    SEB 3,FLAG1                      ;TELEMAC SEND EXT FLAG
1005 1CF7 8D8C1D  STA BUF
1006 1CFA 8D8E03  STA EWDTRP
1007 1CFD A9D6    LDA #0D6H                        ;ENDING ADDR OF VERSION NUMBER
1008 1CFF 8D8D1D  STA BUF2
1009 1D02 60      RTS
1010          ;
1011          MBC_SOFT: ;RTS                          MBC SOFTWARE VERSION, MBC TO
RESPOND
1012          MBCDWG: ;RTS                          MBC HARDWARE VERSION, MBC TO
RESPOND
1013          ;
1014

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

1015          .PAGE
1016          ;
1017          ;*<
1018          ;  NAME:
1019          ;
1020          ;  DESCRIPTION:
1021          ;  CALL:
1022          ;  ARGUMENTS:
1023          ;  MODIFIES:
1024          ;  RETURNS:
1025          ;  HISTORY:
1026          ;*>
1027          ;
1028          LOCK1:
1029 1D03 A901    LDA #1          ;LOCK THE PHONE
1030 1D05 8D411E STA LOCK
1031 1D08 8007    BRA RESTART    ;PWR UP
1032          ;
1033

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```
1034          .PAGE
1035          ;
1036          ;*<
1037          ;  NAME:
1038          ;
1039          ;  DESCRIPTION:
1040          ;  CALL:
1041          ;  ARGUMENTS:
1042          ;  MODIFIES:
1043          ;  RETURNS:
1044          ;  HISTORY:
1045          ;*>
1046          ;
1047 1D0A A900  UNLOCK: LDA #0          ;UNLOCK THE PHONE
1048 1D0C 8D411E STA LOCK
1049 1D0F 8000  BRA RESTART          ;PWR UP
1050          ;
1051
```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

1052          .PAGE
1053          ;
1054          ;*<
1055          ;  NAME:
1056          ;
1057          ;  DESCRIPTION:
1058          ;  CALL:
1059          ;  ARGUMENTS:
1060          ;  MODIFIES:
1061          ;  RETURNS:
1062          ;  HISTORY:
1063          ;*>
1064          ;
1065          RESTART:
1066 1D11 68      PLA
1067 1D12 68      PLA          ;RESTORE STACK, PROBABLY NOT NEEDED
1068 1D13 4C0080  JMP 8000H
1069          ;
1070          CHANGE: ;RTS          CHANGE MANUAL LOCK CODE
1071          ;
1072

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```
1073          .PAGE
1074          ;
1075          ;*<
1076          ; NAME:
1077          ;
1078          ; DESCRIPTION:
1079          ; CALL:
1080          ; ARGUMENTS:
1081          ; MODIFIES:
1082          ; RETURNS:
1083          ; HISTORY:
1084          ;*>
1085          ;
1086          AIR_TIME:                                ;SEND CUMLATIVE AIR TIME
COUNTER
1087 1D16 A91E    LDA #1EH
1088 1D18 8DCD1B  STA EX_PTR                        ;SET UPPER BYTE OF POINTER
1089 1D1B A939    LDA #39H                          ;STARTING ADDR OF AIR TIME (MUST IN OF
HIGH BYTE PAGE)
1090 1D1D 6F2B    SEB 3,FLAG1                        ;TELEMAC SEND EXT FLAG
1091 1D1F 8D8C1D  STA BUF
1092 1D22 8D8E03  STA EWDTRP
1093 1D25 A940    LDA #$38+8                          ;ENDING ADDR OF AIR_TIMER
1094 1D27 8D8D1D  STA BUF2
1095 1D2A 60      RTS
1096          ;
1097
```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

1098          .PAGE
1099          ;
1100          ;*<
1101          ;  NAME:
1102          ;
1103          ;  DESCRIPTION:
1104          ;  CALL:
1105          ;  ARGUMENTS:
1106          ;  MODIFIES:
1107          ;  RETURNS:
1108          ;  HISTORY:
1109          ;*>
1110          ;
1111          CALL_XFR: ;RTS          FORCE PHONE TO CALL TRANSFER
MODE?
1112          NUM_CALLS:          ;PHONE CALL COUNTER
1113 1D2B A91D    LDA #1DH
1114 1D2D 8DCD1B    STA EX_PTR          ;SET UPPER BYTE OF POINTER
1115 1D30 A996    LDA #96H          ;STARTING ADDR OF CALL CNTR (MUST IN
2D HIGH BYTE PAGE)
1116 1D32 6F2B    SEB 3,FLAG1          ;TELEMAC SEND EXT FLAG
1117 1D34 8D8C1D    STA BUF
1118 1D37 8D8E03    STA EWDTRP
1119 1D3A A998    LDA #$96+2          ;ENDING ADDR OF CALL CNTR
1120 1D3C 8D8D1D    STA BUF2
1121 1D3F 60      RTS
1122          ;
1123

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

1124          .PAGE
1125          ;
1126          ;*<
1127          ; NAME:
1128          ;
1129          ; DESCRIPTION:
1130          ; CALL:
1131          ; ARGUMENTS:
1132          ; MODIFIES:
1133          ; RETURNS:
1134          ; HISTORY:
1135          ;*>
1136          ;
1137          PTR:                                ;SEND POINTER
1138 1D40 A91D    LDA #1DH
1139 1D42 8DCD1B  STA EX_PTR                    ;SET UPPER BYTE OF POINTER
1140 1D45 A999    LDA #99H                    ;STARTING ADDR OF CALL PTR (MUST IN OF
HIGH BYTE PAGE)
1141 1D47 6F2B    SEB 3,FLAG1                  ;TELEMAC SEND EXT FLAG
1142 1D49 8D8C1D  STA BUF
1143 1D4C 8D8E03  STA EWDTRP
1144 1D4F A99B    LDA #$9B                    ;ENDING ADDR OF CALL PTR
1145 1D51 8D8D1D  STA BUF2
1146 1D54 60      RTS
1147          ;
1148          PROGRAM: ;RTS                      PROGRAM PHONE, LOCATE IN ROM
1149          NUM_BYTES:
1150          STR_NUM: ;RTS                      SAVE A BYTE, STORE PHONE IN
LOCATION 1-9
1151          ;
1152

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

1153          .PAGE
1154          ;
1155          ;*<
1156          ;  NAME:
1157          ;
1158          ;  DESCRIPTION:
1159          ;  CALL:
1160          ;  ARGUMENTS:
1161          ;  MODIFIES:
1162          ;  RETURNS:
1163          ;  HISTORY:
1164          ;*>
1165          ;
1166          CLR_AIR:          ;RESET CUMULATIVE AIR TIMER
1167 1D55 A900      LDA #0
1168 1D57 A208      LDX #8
1169          ;
1170          LPNOV:
1171 1D59 CA        DEX
1172 1D5A 9D391E    STA ACTIM,X
1173 1D5D D0FA      BNE LPNOV
1174 1D5F 60        RTS
1175

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

1176      .PAGE
1177      ;
1178      ;*<
1179      ;   NAME: Store
1180      ;
1181      ;   DESCRIPTION:
1182      ;           Store a byte of call data in RAM for the current activity
1183      ;
1184      ;   CALL:
1185      ;           INC_UP:      To increment the upper address byte value
1186      ;           MEM_FULL: To Lock the phone when call data ceiling reached
1187      ;
1188      ;   ARGUMENTS:
1189      ;           Byte of data to be stored
1190      ;
1191      ;   MODIFIES:
1192      ;   RETURNS:
1193      ;   HISTORY:
1194      ;           TWW:Aug-09-1991. To modify the call ceiling data
1195      ;*>
1196      ;
1197      ;*****
1198      ;   STORE BYTE (IN ACC) POINTED TO BY C_PTL
1199      ;*****
1200      ;
1201      STORE:
1202 1D60 20981D      jsr      STO_BY      ; STORE BYTE IN ACC
1203 1D63 EE991D      inc      C_PTL      ; Increment pointer low address
byte
1204      ;           beq      INC_UP      ; ROLL OVER?
1205      ;
1206      ;<<<<TWW Aug-09-1991      Test for data ceiling at 0x19EC
1207      ;
1208 1D66 D005      bne      STORE_00      ; TWW:If not Z are we at the
ceiling
1209 1D68 EE9A1D      inc      C_PTL+1      ; TWW:Increment call
data address MSB
1210 1D6B 800E      bra      STORE_EXIT      ; TWW:On to the next
1211      ;
1212      STORE_00:
1213 1D6D A9ED      lda      #0EDH      ; TWW:LSB of call data ceiling
1214 1D6F CD991D      cmp      C_PTL      ; TWW:compare that lower byte
1215 1D72 D007      bne      STORE_EXIT      ; TWW:On to the next
1216 1D74 A919      lda      #19H      ; TWW:MSB of call data ceiling
1217 1D76 CD9A1D      cmp      C_PTL+1      ; TWW:compare that
upper byte
1218 1D79 F001      beq      MEM_FULL      ; TWW:Hit the call ceiling. Lock
Up!!!
1219      ;
1220      ;<<<<TWW
1221      ;
1222      STORE_EXIT:
1223 1D7B 60      rts
1224

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

1225          .PAGE
1226          ;
1227          ;*<
1228          ;  NAME:
1229          ;
1230          ;  DESCRIPTION:
1231          ;  CALL:
1232          ;  ARGUMENTS:
1233          ;  MODIFIES:
1234          ;  RETURNS:
1235          ;  HISTORY:
1236          ;*>
1237          ;
1238          ;INC_UP:
1239          ;      inc      C_PTL+1          ;INC UPPER BYTE OF
POINTER
1240          ;      lda      #1AH          ;OUT OF MEMORY?
1241          ;      cmp      C_PTL+1
1242          ;      beq      MEM_FULL      ; Memory full, Lock it!!!
1243          ;
1244          ;      rts
1245

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

1246          .PAGE
1247          ;
1248          ;*<
1249          ;  NAME:
1250          ;
1251          ;  DESCRIPTION:
1252          ;  CALL:
1253          ;  ARGUMENTS:
1254          ;  MODIFIES:
1255          ;  RETURNS:
1256          ;  HISTORY:
1257          ;*>
1258          ;
1259          MEM_FULL:
1260          ;
1261          ;          seb      6,FLAG15          ;SET MEMORY FULL FLAG
1262          ;          jsr      CLR_AIR
1263 1D7C 4C031D      jmp      LOCK1          ;LOCK PHONE
1264          ;
1265 1D7F 60          rts
1266

```


E SEQ. LOC. OBJ..*....1....*....2....*....SOURCE STATEMENT....5....*.

```

1267          .PAGE
1268          ;*****
1269          *= $1D80
1270 1D80 56312E30 .BYTE 'V1.0 17DEC90'          ;SOFTWARE VERSION NUMBER AND
DATE
          1D84 20313744
          1D88 45433930
1271
1272          *= $1D96
1273 1D96 0000      .WORD $00          ;INIT CALL COUNTER
1274
1275          *= $1D98
1276          STO_BY:
1277 1D98 8D          .BYTE $8D          ;OP CODE STA, FOR SUBROUTINE
STORE
1278 1D99 900F      .WORD $E00+400          ;INIT PHONE # POINTER
1279 1D9B 60          .BYTE $60          ;OP CODE RTS, FOR SUBROUTINE
STORE
1280          ;
1281          ;*****
1282          ;  RECEIVED DATA BYTE FROM MBC (IN ACC)
1283          ;*****
1284          ;  SEE FILE ROMMOD.ASM
1285
1286          .END

```

```

ERROR COUNT 00000
TOTAL LINE 01286 LINES
COMMENT LINE 00799 LINES
OBJECT SIZE 00883 BYTES

```



```
/*-----
convert.c
```

PURPOSE: Converts a sequential 'C' file database to a Delimited Ascii Database.

REQUIRES: A premade template file.

example file template format :

```
*  <- This is a remark
A11
*    a field of chars 11 long
F
*    a float 4 bytes long
A1
*    a char 1 byte long
etc..
*  describes the data base
```

Written By : Greg McGregor 1990

Revised?

What was revised?

GMM 8-7-1991	Changed delimiter to command line option - V1.01
GMM 8-7-1991	Added dates to template file V1.02
GMM 8-11-1991	V1.03 don't worry about padded fields from ascii to c
GMM 8-26-1991	Creates file 'OK' on successful conversion
GMM 8-30-1991	Pause for 2 seconds on successful conversion to see screen
GMM 9-4-1991	V1.07 Byte count fix on converting to C
GMM 9-16-1991	V1.09 Bug in get_ascii_token

-----*/

```
extern unsigned _stklen = 543210;
```

```
#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <dir.h>
#include <conio.h>
#include <io.h>
#include <fcntl.h>
#include <sys\stat.h>
#include <string.h>
```

```
#include <\h2\hdr\gkeys.h>
#include <\h2\hdr\windows.h>
```

```
windef main_win = {10,7,70,12,White,Blue,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
                    White,Blue};
```

```
windef usage_win = {5,3,75,20,White,Red,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
                    White,Red};
```

```
windef title_win = {10,2,70,4,White,Blue,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
                    White,Blue};
```

```
windef action_win = {10,15,70,23,White,Blue,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
                    White,Blue};
```

```
windef done_win = {5,9,75,13,White,Magenta,FALSE,FAI SE,FAI SE,TRUE,SINGLEFRAME,
                    White,Magenta};
```


wintype main_wt,title_wt,action_wt,done_wt,usage_wt;

504

```
char conversion_type[10];
char template_file [80];
char file_name [80];
char new_file [80];
long unsigned recs_converted = 0;
long unsigned size_of_file;
long unsigned number_of_recs;
int template_size;
int number_of_fields =0;
int word_align = TRUE;
```

```
char field_seperator[1];
long unsigned rec_num = 0;
long unsigned bytes_read = 0;
long unsigned address_pointer = 0;
long unsigned bytes_aligned = 0;
long unsigned deleted_records = 0;
```

```
char template [500][80]; /* template file in memory */
char *get_ascii_token(char buff[],int reset_val);
```

```
main (int argc,char *argv[]) {
```

```
    _setcursortype (_NOCURSOR);
```

```
    delay (0);
```

```
    if (argc != 6) {
```

```
        delay (0);
```

```
        /*buzz (); */
```

```
        clrscr ();
```

```
        ruff_area (1,1,80,23,White,Blue);
```

```
        usage_wt = windowopen (&usage_win);
```

```
        setttitle (usage_wt,"Invalid Usage!",CenterUpperTitle);
```

```
        clrscr ();
```

```
        cprintf ("CONVERT");
```

```
        gotoxy (1,2);
```

```
        cprintf ("V1.09");
```

```
        gotoxy (1,4);
```

```
        cprintf ("USAGE: Convert [To Type] [Template] [File Name] [New Name]");
```

```
        gotoxy (1,6);
```

```
        cprintf ("          Required:");
```

```
        gotoxy (1,7);
```

```
        cprintf ("          To Type    - A = Convert To ASCII, C = Convert To
```

```
        gotoxy (1,8);
```

```
        cprintf ("          Template  - Template File Name");
```

```
        gotoxy (1,9);
```

```
        cprintf ("          File Name - File Name To Convert");
```

```
        gotoxy (1,10);
```

```
        cprintf ("          New Name  - File To Produce, Output");
```

```
        gotoxy (1,11);
```

```
        cprintf ("          Field Seperator - 0 = ' ', 1 = '|' , 2 = '.'");
```

```
        gotoxy (1,13);
```

```
        cprintf ("PURPOSE: Converts From ASCII DataBases To 'C' DataBases");
```

```
        gotoxy (1,14);
```

```
        cprintf ("          And Vica Versa");
```

```
        gotoxy (1,15);
```

```
        cprintf ("GMM 1991");
```

```
        _setcursortype (_NORMALCURSOR);
```

```
        window (1,1,80,25);
```

```
        textcolor (White);
```



```

        textbackground (Black);
        gotoxy (1,24);
        exit (0);
    }
    strcpy (conversion_type,argv[1]);
    strcpy (template_file,argv[2]);
    strcpy (file_name,argv[3]);
    strcpy (new_file,argv[4]);
    if (argv[5][0] == '0') {
        field_seperator[0] = ',';
    } else
    if (argv[5][0] == '1') {
        field_seperator[0] = '|';
    } else
    if (argv[5][0] == '2') {
        field_seperator[0] = '.';
    } else {
        clrscr ();
        printf ("ERROR: In field seperator. Valid inputs are 0,1,2");
        exit (0);
    }
    init_template ();
    start_process ();
}

```

```

/*-----
ahoh
-----*/

```

```

void ahoh (void)

```

```

{
    sound (200);
    delay (150);
    nosound();
    delay (20);
    sound (150);
    delay (250);
    nosound ();
}

```

```

/*-----
init_template
-----*/

```

```

init_template () {
    int i;
    int j;
    for (i = 0;i<500;i++)
        for (j=0;j<10;j++)
            template[i][j] = '\0';
}

```

```

/*-----
start_process
-----*/

```

```

start_process () {
    ruff_area (1,1,80,25,White,Blue);
    title_wt = windowopen (&title_win);
    setttitle (title_wt,"Convert - Version 1.09",CenterUpperTitle);
    clrscr ();
    cprintf ("

```

```

        C O N V E R T");

```



```

main_wt = windowopen (&main_win);
settitle (main_wt,"Process",CenterUpperTitle);
action_wt = windowopen (&action_win);
settitle (action_wt,"Action",CenterUpperTitle);
if ((conversion_type[0] == 'A') || (conversion_type[0] == 'a') )
    do_process_ascii ();
if ((conversion_type[0] == 'C') || (conversion_type[0] == 'c') )
    do_process_c ();

```

```

close_all_windows ();
window (1,1,80,25);
textbackground (Black);
textcolor (White);
clrscr ();
printf ("\n ERROR : In Command Line Parameters!");
}

```

```

/*-----
null_data
-----*/

```

```

null_data (char *s,int len) {
    int i;
    for (i=0;i<len;i++)
        s[i] = '\0';
}

```

```

/*-----
pad_field
-----*/

```

```

pad_field (char *s, int len) {
    int i,j;
    i = 0;
    while (s[i] != '\0') ++i;
    for (j=i;j<len-1;j++)
        s[j] = ' ';
}

```

```

/*-----
calc_size
-----*/

```

```

calc_size () {
    int i;
    char temp[80];

    number_of_fields = 0;
    template_size = 0;
    i = 0 ;
    while (template[i][0]) {
        strcpy (temp,template[i]);
        if (temp[0] == 'A') {
            temp[0] = '0';
            template_size += atoi (temp);
            ++number_of_fields;
        }
        if (temp[0] == 'F') {
            template_size += sizeof (float);
            ++number_of_fields;
        }
        ++i;
    }
    use (action_wt);
    gotoxy (1,3);
    cprintf ("Record Size      -> %d",template_size);
}

```



```
do_process_c () {
int i,new_line,j;
char s[10000];
int pos = 0;
int value;
FILE *fd;
int fd1;
char buff [10000],ch;
int stat;
float fl;
char temp[255],temp2[255],t[80];
char *p;
int ok,end_of_record;
long unsigned records_processed = 0;
long unsigned unused_bytes = 0;
long unsigned bytes_produced = 0;
int actual_number_of_fields = 0;

    fd = fopen (file_name,"rb+");
    if (fd == NULL)
        quit (-4);
    fd1 = open (new_file,O_CREAT | O_BINARY | O_WRONLY | O_TRUNC,S_IWRITE);
    if (fd1 == -1 )
        quit (-3);
    show_tech ();
    read_in_template ();
    calc_size ();

    gotoxy (30,4);
    cprintf ("Bytes Produced      -> %lu",bytes_produced);
    gotoxy (1,6);
    if (word_align)
        cprintf ("Word Alignment [ON]");
    if (!word_align)
        cprintf ("Word Alignment [OFF]");

    null_data (buff,10000);
    actual_number_of_fields = 0;
    while (fgets (buff,10000,fd) ) {
        actual_number_of_fields = 0;
        /* format_line (buff); */
        temp[0] = 1;
        temp[1] = '\0';
        new_line = TRUE;
        i = 0;
        address_pointer = 0;
        ok = TRUE;
        end_of_record = FALSE;
        while (template[i][0] != '\0') {
            if (ok) {
                null_data (temp,255);
                null_data (temp2,255);
                if (new_line) {
                    p = get_ascii_token (buff,TRUE);
                    new_line = FALSE;
                } else p = get_ascii_token (buff,FALSE);
                strcpy (temp,p);
                ok = FALSE;
                ++actual_number_of_fields;
            }
            if (template[i][0] == 'A') {
                strcpy (temp2,template[i]);
                temp2[0] = '0';
                value = atoi (temp2);
            }
        }
    }
}
```



```

        stat = write (fd1,temp,value);
        if (stat == -1)
            quit (-6);
        address_pointer += value;
        bytes_produced += value;
        ok = TRUE;
    }
    /*treat dates as strings NO FORMATING */
    if (template[i][0] == 'D') {
        null_data (t,80);
        j = 1; /* get field length */
        while (template[i][j] != ':') {
            t[j-1] = template[i][j];
            ++j;
            if (j>50) quit (-2,i);
        }

        value = atoi (t);
        address_pointer += value;
        bytes_produced += value;
        if (stat == -1)
            quit (-5);
        ++j;
        null_data (t,80);
        pos = 0;
        while (template[i][j] != ':') {
            switch (template[i][j]){
                case 'M':
                    t[2] =buff[pos++];
                    t[3] =buff[pos++];
                    break;
                case 'Y':
                    t[0] =buff[pos++];
                    t[1] =buff[pos++];
                    break;
                case 'D':
                    t[4] =buff[pos++];
                    t[5] =buff[pos++];
                    break;
            }
            ++j;
        }
        stat = write (fd1,t,value);
        ok = TRUE;
    }
    if (template[i][0] == 'F' ) {
        if ( (address_pointer % 2) != 0) {
            if (word_align) {
                ch = ' ';
                stat = write (fd1,&ch,1);
                if (stat == -1)
                    quit (-6);
                ++address_pointer;
                ++unused_bytes;
            }
        }
        fl = (float) atof (temp);
        stat = write (fd1,&fl,sizeof (float));
        if (stat == -1)
            quit (-6);
        value = sizeof (float);
        address_pointer += value;
        bytes_produced += sizeof (float);
        ok = TRUE;
    }
    if (strncmp (template[i],"WORD ON",7) == 0)

```



```

        word_align = TRUE;
        if (strncmp (template[i], "WORD OFF", 8) == 0)
            word_align = FALSE;
        ++i;
    }
    if ( (address_pointer % 2) != 0) {
        ch = ' ';
        ++address_pointer;
        ++unused_bytes;
        stat = write (fd1, &ch, 1);
        if (stat == -1)
            quit (-6);
        ++bytes_produced;
    }
    null_data (buff, 10000);
    new_line = TRUE;
    ++records_processed;
    use (action_wt);
    gotoxy (30, 2);
    cprintf ("Records converted -> %lu", records_processed);
    gotoxy (30, 3);
    cprintf ("Record Size      -> %lu", address_pointer);
    gotoxy (1, 4);
    ++actual_number_of_fields;
    cprintf ("Number of Fields  -> %d", number_of_fields);
    if (word_align) {
        gotoxy (1, 7);
        cprintf ("Unused Bytes Used For Word Alignment -> %lu", unused_bytes);
    } else {
        gotoxy (1, 7);
        cprintf ("No Unused Bytes! (Compact Data)");
    }
    gotoxy (30, 4);
    cprintf ("Bytes Produced      -> %lu", bytes_produced);
}

number_of_recs = records_processed;
deleted_records = 0;
fclose (fd);
close (fd1);
quit (1);
}

/*-----
format_line : reset field delimiators to hex 1
-----*/
format_line (char *s) {
    int i, pos, k, j;
    int val;
    float fl;
    char temp [100];

    i = 0;
    pos = 0;
    while (template[i][0] != '\0') {
        if (template[i][0] == 'A') {
            strcpy (temp, template[i]);
            temp[0] = '0';
            val = atoi (temp);
            pos += val;
            j = pos;
            while ( (s[j] != field_seperator[0]) && (s[j] != '\n') )
                ++j;
            s[j] = 0x1B; /* reset field delimiter to hex 1B ESC */
            pos += j - pos;
            ++pos; /* start of next field */
        }
        ++i;
    }
}

```



```

        if (template[i][0] == 'F') {
            pos += sizeof (float);
            j = pos;
            while ( (s[j] != field_seperator[0]) && (s[j] != '\n') )
                ++j;
            s[j] = 0x18;
            pos += j-pos;
            ++pos;
        }
        ++i;
    }
    s[pos] = 0x1C; /* EOL char */
}

```

 /*-----
 get_ascii_token
 -----*/

```

char *get_ascii_token(char *buff,int reset_val) {
    static int pos = 0;
    char data[500];
    int i;

    if (reset_val) /* reset if requested */
        pos = 0;
    i = 0;
    null_data (data,500);
    while ( (buff[pos] != field_seperator[0]) && (buff[pos] != '\n') &&
            (buff[pos] != '\0') ) {
        if (buff[pos] != '\n') {
            data[i] = buff[pos];
            ++i;
        }
        ++pos;
    } /* skip space between two records */
    /* if (buff[pos] != field_seperator[0]) {
        i = 0;
        null_data (data,500);
        while (buff[pos] != field_seperator[0])
            ++pos;
        ++pos; /* to get past the field seperator */
        while (buff[pos] != field_seperator[0]) {
            data[i] = buff[pos];
            ++i;
            ++pos;
        }
    }

    ++pos; /* to get past delimiter */
    /* if (buff[pos] == 0x18) ++pos; /* get ready for next token */
    if (buff[pos] == 0x1C)
        data[0] = 0x1C; /* signal end of record */

    /*
    return data;
}

```

 /*-----
 null_to_space
 -----*/

```

null_to_space (char *s,int pos,int run) {
    int i;
    for (i=pos;i<=pos+run;i++) {
        if (s[i] == '\0')
            s[i] = ' ';
    }
}

```



```
/*-----  
do_process_ascii  
-----*/  
do_process_ascii () {  
    int i,j;  
    char s[10000];  
    int pos = 0;  
    FILE *fd;  
    int value;  
    int fd1;  
    char buff [1000],ch;  
    int stat;  
    float fl;  
    char temp[80];  
    long unsigned bytes_produced = 0;  
    char t[80],t1[80];  
  
        fd = fopen (new_file,"wb+");  
        if (fd == NULL) {  
            quit (-3);  
        }  
        fd1 = open (file_name,O_RDONLY | O_BINARY, S
```



```

/*-----
do_process_ascii
-----*/

do_process_ascii () {
int i,j;
char s[10000];
int pos = 0;
FILE *fd;
int value;
int fd1;
char buff [1000],ch;
int stat;
float fl;
char temp[80];
long unsigned bytes_produced = 0;
char t[80],t1[80];

    fd = fopen (new_file,"wb+");
    if (fd == NULL) {
        quit (-3);
    }
    fd1 = open (file_name,O_RDONLY | O_BINARY, S_IREAD);
    if (fd1 == -1 )
        quit (-4);

    show_tech ();
    read_in_template ();
    calc_size ();

    size_of_file = filelength (fd1);
    if (size_of_file == -1 )
        quit (-5);
    number_of_recs = size_of_file / template_size;
    use (action_wt);
    gotoxy (30,3);
    cprintf ("Number of Records -> %lu",number_of_recs);
    gotoxy (1,4);
    cprintf ("Number of Fields -> %d",number_of_fields);
    gotoxy (30,4);
    cprintf ("Bytes Produced -> %lu",bytes_produced);
    gotoxy (1,6);
    if (word_align)
        cprintf ("Word Alignment [ON]");
    if (!word_align)
        cprintf ("Word Alignment [OFF]");

loop: while (rec_num < number_of_recs) {
    null_data (s,10000);

    i = 0;
    pos = 0;
    address_pointer = 0;
    while (template[i][0] != '\0') {
        if (template[i][0] == 'A') {
            /*
            s[pos] = ' ';
            ++pos;

```



```

*/
strcpy (temp,template[i]);
temp[0] = '0';
value = atoi (temp);
null_data (buff,1000);
stat = read (fd1,buff,value);
bytes_read += stat;
if (stat == -1)
    quit (-5);
strcat (s,buff);
null_to_space (s,pos,value); /* zap out nulls in fi
pos += value;
/*
s[pos] = ' ';
++pos;
*/
address_pointer += stat;
bytes_produced += value;
}
if (template[i][0] == 'F') {
    if ( ( (address_pointer % 2) != 0) &&
        (word_align) ) { /* word alignment */
        stat = read (fd1,&ch,1); /* unused byte */
        ++bytes_read;
        ++address_pointer;
        ++bytes_aligned;
    }
    stat = read (fd1,&f1,sizeof (float)); /* 4 is sizeof
bytes_read += stat;
address_pointer += stat;
if (stat == -1)
    quit (-5);
sprintf (temp,"%0.2f",f1);
strcat (s,temp);
value = strlen (temp);
pos += strlen (temp);
bytes_produced += value;
}
if (template[i][0] == 'D') {
    null_data (t,80);
    j = 1; /* get field length */
    while (template[i][j] != ':') {
        t[j-1] = template[i][j];
        ++j;
        if (j>50) quit (-2,i);
    }

    value = atoi (t);

    stat = read (fd1,buff,value);
    bytes_read += stat;
    address_pointer += stat;
    if (stat == -1)
        quit (-5);
    ++j;
    while (template[i][j] != ':') {
        switch (template[i][j]){
            case 'M':
                s[pos++] =buff[2];
                s[pos++] =buff[3];
                break;
            case 'Y':
                s[pos++] =buff[0];
                s[pos++] =buff[1];
                break;
            case 'D':

```


s[pos++] = buff[4];
s[pos++] = buff[5];
break; 513

```

    }
    ++j;
}
}
if ( (template[i+1][0] != '\0') && (template[i][0] != '*') &
    (template[i][0] != 'W') ){
    s[pos] = field_seperator[0];
    ++pos;
}
if (s[0] == '~') { /* don't write deleted records */
    ++deleted_records;
    goto loop;
}
/*
    if (s[pos-1] == field_seperator[0]) { /*
        s[pos-1] = ''; /* the end, za
        s[pos] = '\0';
    }
    fprintf (fd,"%s\n",s);
    bytes_produced += strlen (s);
*/
}
if (template[i+1][0] == '\0') {
    if (word_align) {
        if ( (address_pointer % 2) != 0) {
            stat = read (fd1,&ch,1);
            ++bytes_read;
            ++address_pointer;
            ++bytes_aligned;
        }
    }
}
++i;
}
fprintf (fd,"%s\n",s); /* write out data to disk */
use (action_wt);
gotoxy (30,2);
++rec_num;
cprintf ("Records converted -> %lu",rec_num);
gotoxy (30,4);
cprintf ("Bytes Produced -> %lu",bytes_produced);
if (word_align) {
    gotoxy (1,7);
    cprintf ("Unused Bytes Used For Word Alignment -> %lu",bytes
} else {
    gotoxy (1,7);
    cprintf ("No Unused Bytes! (Compact Data)");
}
gotoxy (30,5);
cprintf ("Deleted Records -> %lu",deleted_records);
}
fclose (fd);
close (fd1);
quit (1);
}

```

```

/*-----
shift_string
-----*/
shift_string (char *s,int len) {
int i,j;
    for (j=0;j<len;j++)
        for (i=0;i<strlen (s);++i) {
            s[i] = s[i+1];

```



```

}

/*-----
show_tech
-----*/
show_tech () {
char s[80];
    if ((conversion_type[0] == 'A') || (conversion_type[0] == 'a')) {
        strcpy (s,"ASCII DELIMITED");
    } else
        strcpy (s,"Sequential 'C'");
    use (main_wt);
    gotoxy (5,1);
    cprintf ("Converting To -> %s",s);
    gotoxy (5,2);
    cprintf ("File Name      -> %s",file_name);
    gotoxy (5,3);
    cprintf ("Template Name  -> %s",template_file);
    gotoxy (5,4);
    cprintf ("Producing File -> %s",new_file);
    use (action_wt);
    gotoxy (30,2);
    cprintf ("Records Converted -> %lu",recs_converted);
}

```

```

/*-----
read_in_template ()
-----*/
read_in_template () {
int i;
FILE *fd;
char line[80];

    fd = fopen (template_file,"rb");
    if (fd == NULL)
        quit (-1);
    i = 0;
    while (fgets (line,80,fd) ) {
        strcpy (template[i],line);
        ++i;
    }
    fclose (fd);
    quick_check (); /* quickly check format, parse it */
}

```

```

/*-----
quick_check
-----*/
quick_check () {
int i;

        /* a '*' is a rem statement */

    i = 0;
    while (template[i][0]) {
        if ( (template [i][0] != 'A') &&
            (template [i][0] != 'F') &&
            (template [i][0] != '*') &&
            (template [i][0] != 'W') &&
            (template [i][0] != 'D' )) {
            quit (-2,++i);
        }
        if (strncmp (template [i],"WORD ON",7) == 0) {
            word_align = TRUE;
        }
    }
}

```



```

    }
    if (strcmp (template [i], "WORD OFF", 8) == 0) {
        word_align = FALSE;
    }
    ++i;
    use (action_wt);
    gotoxy (1, 2);
    cprintf ("Template Syntax [%d]", i);
}
use (action_wt);
gotoxy (1, 2);
cprintf ("Template Syntax [%d] -OK", i);
}

```

```

/*-----
quit
-----*/

```

```

quit (int id, int location) {
    int i;
    int j, fd;
    int count = 0;
    int done;
    _setcursortype (_NORMAL_CURSOR);
    window (1, 1, 80, 25);
    textcolor (White);
    textbackground (Black);
    switch (id) {
        case -1:
            clrscr ();
            printf ("\nERROR : Can't Find Template File %s", tem_
            exit (0);
            break;

        case -2:
            clrscr ();
            printf ("\n ERROR : Syntax Error Line (%d) in Templ
            exit (0);
            break;

        case -3:
            clrscr ();
            printf ("\n ERROR : Can't Open Output File %s", new_
            exit (0);
            break;

        case -4:
            clrscr ();
            printf ("\n ERROR : Can't Open Source File %s", file
            exit (0);
            break;

        case -5:
            clrscr ();
            printf ("\n ERROR : Reading Source File %s", file_na
            exit (0);
            break;

        case -6:
            clrscr ();
            printf ("\n ERROR : Writing to Output File %s", new_
            exit (0);
            break;

        case -7:
            clrscr ();
            printf ("\n ERROR : # of fields in template don't ma
            exit (0);
            break;

        case 1:

```

```

/*

```



```

buzz ();
*/
done = FALSE;
done_wt = windowopen (&done_win);
settitle (done_wt, "DONE!", CenterUpperTitle);
clrscr ();
cprintf ("          %s -> %s    - Completed!", file_na
gotoxy (1,2);
cprintf ("Total Records %lu, Deleted Records %lu, P
                                deleted_records, number_of_re
/* gotoxy (1,3);
cprintf ("                                PRESS <ANY> KEY TO
while (!done) {
    for (i=0; i<2000; i++) {
        if (kbhit ()) {
            getch ();
            done = TRUE;
        } else delay (1);
    }
    buzz ();
}
*/
delay (2000);

close_all_windows ();
window (1,1,80,25);
textcolor (White);
textbackground (Black);
clrscr ();
printf ("\nThank you....GMM 1991");
fd = open ("OK", O_CREAT|O_WRONLY, S_IWRITE);
close (fd);
exit (0);
break;
}
clrscr ();
printf ("\n ERROR : Genera); " Failure!");
exit (0);
}

/*-----
buzz
-----*/
buzz () {
int i;
    for (i=1; i<100; i++) {
        sound (1000);
        delay (1);
        nosound ();
        delay (1);
    }
}

```



```
/*-----
transfer V1.56t
```

PURPOSE:

Downloads a hotels data.
 Performs various functions, date time check etc...

Written By: Greg McGregor 1990

Modified: GMM 7-2-1991

REVISED:

What was revised?

GMM 8-14-1991

Zap and execute files.
 Press ESC to exit on connect
 other misc.
 get phone number from a file

GMM 8-26-1991

creates file 'OK' on successful transfer

GMM 9-19-1991

got rid of g,p commands. Deleted 'OK' at start of transfer
 allows for zero length file transfers. Some extra error che
 the commands line argument.

-----*/

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <process.h>
#include <time.h>
#include <window.h>
#include <math.h>
#include <float.h>
#include <dos.h>
#include <bios.h>
#include <fcntl.h>
#include <sys\stat.h>
#include "asiports.h"
#include "xfer.h"
#include "ibmkeys.h"
#include "gf.h"
```

```
#include <\h2\gmmlib\gmmlib.h> /* greg's lib */
```

```
void status_routine (char *m);
void transfer_status (XFER *b);
char calc_CRC (char *s,int len);
char send_command (char c);
char recieve_command (char c);
```

```
/*
```

```
* Window Defs
```

```
*/
```

```
windef main_win = {1,1,80,24,White,Blue,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
                  White,Blue};
windef status_win = {10,9,70,16,White,Blue,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
                  White,Blue};
windef error_win = {10,10,70,15,White,Red,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
                  White,Red};
windef import_win = {10,5,70,15,White,Black,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
                  White,Black};
```



```

/*
 * Window Types
 */
wintype main_wt, status_wt, error_wt, import_wt;

#define FULL 1
#define HALF 0
#define MODE ASINOUT|BINARY|NORMALRX
#define RXLEN 2000
#define TXLEN 2000
#define SECONDS 2
#define TRUE 1
#define FALSE 0
#define ECHO 0
#define SPEAKER OFF

int ACK_CHAR = 0x20;
int NAK_CHAR = 0x21;
int LOG_OUT = 0x22;
int SEND_COMMAND = 0x23; /* char send by server saying Iam ready */

int PORT;
int BAUD = 2400; /* Hotels are all at 2400 Baud */
int PARITY = P_NONE; /* No Parity */
char PHONE_NUMBER [80]; /* phone number to call */
int STOP_BITS = 1;
int WORD_LENGTH = 8;
int DUPLEX = FULL;
char command_list[80];
char file_name[80];
int file_number;
char files[10][80]; /* upto 10 files */
char *valid_commands = "aAdDfFlLuUxXyYrRsSnN";

/*-----
// Function Name -> does_file_exist
// Parameters:
// Function:
// Returns:
// Written By : Greg McGregor
//
-----*/
int does_file_exist (char *f) {
int fd;
fd = open(f,O_RDONLY|O_BINARY,S_IREAD);
if (fd < 0) return ( FALSE );
close (fd);
return ( TRUE );
}

/*-----
main :
-----*/
main (int argc,char *argv[])
{
int i;
FILE *fp;
if (does_file_exist ("ok. ")) system ("del ok. >trash"); /* remove ok
init_windows ();
check_args (argc,argv);
PORT = atoi (argv[1]) - 1; /* set Port */
fp = fopen (argv[2],"r");
if (fp == NULL) {

```



```

    clrscr ();
    printf ("\nERROR: Can't open file '%s'",argv[2]);
    exit (0);
}

```

```

fscanf (fp,"%s",PHONE_NUMBER);
fclose (fp);

```

```

strcpy (command_list,argv[3]);
strcpy (file_name,argv[4]);
file_number = 0;
i = 4;
for (i=4;i<argc;i++){
    strcpy (files[file_number],argv[i]);
    ++file_number;
}
file_number = 0;
connect_to_site ();
close_all_windows ();
window (1,1,80,25);
textcolor (White);
textbackground (Black);
windowclose (status_wt);
clrscr ();
}

```

```

/*-----
// Function Name -> is_char_in_string
// Parameters:
// Function:
// Returns:
// Written By : Greg McGregor
//
-----*/

```

```

int is_char_in_string (char c,char *s) {
char *t;
    t = s;
    while ( *t ) {
        if (c == *t) return ( TRUE );
        ++t;
    }
    return ( FALSE );
}

```

```

/*-----
// Function Name -> are_commands_valid
// Parameters:
// Function:
// Returns:
// Written By : Greg McGregor
//
-----*/

```

```

int are_commands_valid (char *commands) {
char *list;
    list = commands;
    while (*list) {
        if (!is_char_in_string (*list,valid_commands)) return (FALSE);
        ++list;
    }
    return (TRUE);
}

```

```

/*-----
check_args : check command line args for validity
-----*/

```



```

check_args (int n,char *l[])
{
    if (n < 4){
        main_wt = windowopen (&main_win);
        setttitle (main_wt,"How 'TRANSFER' Works",CenterUpperTitle);
        clrscr ();
        printf ("TRANSFER  V1.56");
        gotoxy (1,2);
        printf ("(GVN Network)");
        gotoxy (1,3);
        printf ("USAGE:  transfer [PORT] [FILE] [COMMANDS] <f|FILE>");
        gotoxy (1,4);
        printf ("      Required: ");
        gotoxy (1,5);
        printf ("      [PORT]      - 1 .. 4   (COM1 through COM4)");
        gotoxy (1,6);
        printf ("      [FILE]      - Contains A Phone Number To Call ");
        gotoxy (1,7);
        printf ("      [COMMANDS]  a - Archive Data Base At TAU");
        gotoxy (1,8);
        printf ("                  d - Date/Time Setting");
        gotoxy (1,9);
        printf ("                  f - File Operation Follows");
        gotoxy (1,10);
        printf ("                  l,u - Lock/Unlock Site");
        gotoxy (1,11);
        printf ("                  x,y - Data Lock/Unlock Data");
        gotoxy (1,12);
        printf ("                  r,s - Reboot/Set life Span");
        gotoxy (1,13);
        printf ("                  n - Get Site's Serial Number");
        gotoxy (1,16);
        printf ("      Optional :  If 'f' is Specified in Commands");
        gotoxy (1,17);
        printf ("      <f|FILE>  -  f = Flag, FILE = File Name");
        gotoxy (1,18);
        printf ("                  FLAGS:");
        gotoxy (1,19);
        printf ("                  -s = Send A File");
        gotoxy (1,20);
        printf ("                  -r = Retrieve A File");
        gotoxy (1,21);
        printf ("                  -e,-z = Execute/Zap a File");
        gotoxy (1,22);
        printf ("GMM 1991");
        window (1,1,80,25);
        gotoxy (1,24);
        exit (0);
    }
    if (!are_commands_valid (l[3]) ) {
        clrscr ();
        printf ("\nERROR: The commands string contains and invalid command!");
        exit (0);
    }
}

/*-----
error :
-----*/
error (int e)
{
    char message [80];

    sprintf (message,"ERROR  %d",e);
    switch (e) {
        case -2 : sprintf (message,"Invalid Port! %d",e);

```



```

        break;
    case -3 : sprintf (message,"Port Already Inuse! %d",e);
        break;
    case -4 : sprintf (message,"Invalid Buffer Size! %d",e);
        break;
    case -5 : sprintf (message,"Memory Allocation Error In Port Setup! %d",e);
        break;
    case -6 : sprintf (message,"Port Not Setup! %d",e);
        break;
    case -7 : sprintf (message,"Invalid Parameter! %d",e);
        break;
    case -23 : sprintf (message,"Modem Not Responding! %d",e);
        break;
    case -22 : sprintf (message,"Modem Not Responding! %d",e);
        break;
    case -100: sprintf (message,"Can't Reset Modem! %d",e);
        break;
}
error_wt = windowopen (&error_win);
settitle (error_wt,"ERROR",CenterUpperTitle);
gotoxy (1,2);
cprintf ("%s",message);
hang_up ();
exit (0);
}

```

```

/*-----
hang_up
-----*/

```

```

hang_up ()
{
    int i;
    cprintf ("    -* Closing Port");
    while (!istxempty (PORT) );
    timer (TICKS_PER_SECOND + 1);
    asiputs (PORT,"+++",-1);
    while (!istxempty (PORT) );
    timer (TICKS_PER_SECOND * 2);
    HMSetHookSwitch (PORT, ONHOOK);
    asiquit (PORT);
}

```

```

/*-----
open_port:
-----*/

```

```

open_port ()
{
    int stat;

    stat = ASSUCCESS;
    clrscr ();
    cprintf ("-* Opening Port");
    if ((stat = asifirst (PORT,MODE,RXLEN,TXLEN)) < ASSUCCESS){
        error (stat);
    }
    if ((stat = asiinit(PORT,BAUD,PARITY,STOP_BITS,WORD_LENGTH))
        < ASSUCCESS ) {
        error (stat);
    }
    if ( (stat = asdtr(PORT,ON)) < ASSUCCESS)
        error (stat);
    if ( (stat = asrts (PORT,ON)) < ASSUCCESS)
        error (stat);
    if ( (stat = asistart(PORT,ASINOUT)) < ASSUCCESS)
        error (stat);
}

```



```

/*-----
init_modem : initialize modem
-----*/
init_modem ()
{
    int stat,i;

    use (status_wt);
    clrscr();
    cprintf ("-* Initializing Modem...");
    HMWaitForOK (TICKS_PER_SECOND*SECONDS,NULL);
    HMSetUpAbortKey (ESC);
    i = 0;
    stat = HMReset (PORT);
    while ( (stat < ASSUCCESS) && (i < 4) ){
        ++i;
        stat = HMReset (PORT);
        gotoxy (1,i+1);
        cprintf ("-* Trying To Reset Modem Again!");
        hang_up ();
        open_port ();
        HMWaitForOK (TICKS_PER_SECOND*SECONDS,NULL);
        HMSetUpAbortKey (ESC);
    }
    if (stat < ASSUCCESS)
        error (stat);
    if (ECHO == 0)
        if ( (stat = HMSetEchoMode (PORT,OFF)) < ASSUCCESS) /* set e
            error (stat);
    if (ECHO == 1)
        if ( (stat = HMSetEchoMode (PORT,ON)) < ASSUCCESS)
            error (stat);
    if ( (stat = HMSetVerboseMode (PORT,ON)) < ASSUCCESS)
        error (stat);
        /* verbal response */
    if ( (stat = HMSetFullDuplexMode (PORT,ON)) < ASSUCCESS) /* duplex FU
        error (stat);
    if ( (stat = HMSetSpeaker (PORT,SPEAKER)) < ASSUCCESS) /* set speaker
        error (stat);
}

/*-----
connected : PREDICATE is connected to site
-----*/
int connected ()
{
    return iscd (PORT,CUMULATIVE);
}

/*-----
start_commands: main processing loop
-----*/
start_commands ()
{
    int command,fd,jumped_out = FALSE;
    int stat,i,trys;
    int command_stat = FALSE;
    int ok = TRUE;

    if (asiclear (PORT,ASINOUT) < 0) {
        cprintf ("Can't Clear Buffer!");
    }
}

```



```

command = 0;
trys = 0;
while (command_list[command]) {
    clrscr ();
    cprintf ("-* Waiting For Job Request %d", trys);
    trys = 0;
    while (!(stat = recieve_command (SEND_COMMAND))) {
        clrscr ();
        cprintf ("-* Waiting For Job Request %d", trys);
        ++trys;
        if (trys > 1) {
            jumped_out = TRUE;
            goto shit; /* break and do log out, 2 trys */
        }
    }
    if ( (command_list[command] == 'f') ||
        (command_list[command] == 'F') )
        command_stat = file_transfer ();
    if ( (command_list[command] == 'd') ||
        (command_list[command] == 'D') )
        command_stat = date_check ();
    if ( (command_list[command] == 'a') ||
        (command_list[command] == 'A') )
        command_stat = archive_database ();
    if ( (command_list[command] == 'l') ||
        (command_list[command] == 'L') )
        command_stat = lock_site ();
    if ( (command_list[command] == 'u') ||
        (command_list[command] == 'U') )
        command_stat = unlock_site ();
    if ( (command_list[command] == 'r') ||
        (command_list[command] == 'R') )
        command_stat = reboot_site ();
    if ( (command_list[command] == 'n') ||
        (command_list[command] == 'N') )
        command_stat = get_serial_number ();
    if ( (command_list[command] == 's') ||
        (command_list[command] == 'S') )
        command_stat = set_life_span ();
    if ( (command_list[command] == 'x') ||
        (command_list[command] == 'X') )
        command_stat = data_lock ();
    if ( (command_list[command] == 'y') ||
        (command_list[command] == 'Y') )
        command_stat = unlock_data ();
    if (!command_stat) ok = FALSE;
    ++command;
}
clrscr ();
cprintf ("-* Getting ready for LOG OUT!");
trys = 0;
while (!recieve_command (SEND_COMMAND)) {
    ++trys;
    if (trys > 1) break; /* break and do log out */
}
shit:
clrscr ();
cprintf ("-* Logging out");
for (i=1; i<4; i++){ /* send 3 log messages */
    while ( (stat = send_xchar (LOG_OUT)) != ASSUCCESS) ; /* log out */
    timer (TICKS_PER_SECOND);
}
if ( (!jumped_out) && (ok) ){
    fd = open ("OK", O_CREAT|O_WRONLY, S_IWRITE);
    close (fd);
}
}

```



```
/*-----  
data_lock  
-----*/  
int data_lock () {  
    clrscr ();  
    cprintf ("-* Data Locking Site!");  
    if (!send_command (0x11) )  
        return FALSE;  
    gotoxy (1,2);  
    cprintf ("-* Site is Now Data Locked!");  
    return TRUE;  
}  
  
/*-----  
unlock_data  
-----*/  
int unlock_data () {  
    clrscr ();  
    cprintf ("-* Unlocking Data Lock At Site!");  
    if (!send_command (0x12) )  
        return FALSE;  
    gotoxy (1,2);  
    cprintf ("-* Site is Now Data Unlocked!");  
    return TRUE;  
}  
  
/*-----  
lock_site  
-----*/  
int lock_site ()  
{  
    clrscr ();  
    cprintf ("-* Locking Site");  
    if (!send_command (0x0B) )  
        return FALSE;  
    gotoxy (1,2);  
    cprintf ("-* Site is Now Locked!");  
    return TRUE;  
}  
  
/*-----  
unlock_site  
-----*/  
int unlock_site ()  
{  
    clrscr ();  
    cprintf ("-* Attempting to Unlock Site");  
    if (!send_command (0x0E) )  
        return FALSE;  
    gotoxy (1,2);  
    cprintf ("-* Site Successfully Unlocked!");  
    return TRUE;  
}  
  
/*-----  
reboot_site  
-----*/  
int reboot_site ()  
{  
    clrscr ();  
    cprintf ("-* Attempting to reboot site");  
    if (!send_command (0x0A) )  
        return FALSE;  
}
```



```

gotoxy (1,2);
cprintf ("-* Site was rebooted!");
return TRUE;
}

```

```

/*-----
get_serial_number
-----*/

```

```

int get_serial_number () {
int stat;
    clrscr ();
    cprintf ("-* Requesting Serial # of site");
    if (!send_command (0x0C) )
        return FALSE;
    clrscr ();
    cprintf ("-* Retrieving Serial # in file 'serial.dat'");
    stat = YmodemReceive (PORT,status_routine,NULL,ESC);
    return TRUE;
}

```

```

/*-----
set_life_span
-----*/

```

```

int set_life_span () {
    clrscr ();
    cprintf ("-* Attempting to set life span");
    if (!send_command (0x0D) )
        return FALSE;
    gotoxy (1,2);
    cprintf ("-* Life span SET!");
    return TRUE;
}

```

```

/*-----
file_transfer
-----*/

```

```

int file_transfer ()
{
int stat;
    strcpy (file_name,files[file_number]);
    if ( (strcmp (file_name,"-r",2) == 0) ||
        (strcmp (file_name,"-R",2) == 0) ) {
        stat = get_file ();
        ++file_number;
    } else
    if ( (strcmp (file_name,"-s",2) == 0) ||
        (strcmp (file_name,"-S",2) == 0) ) {
        stat = send_file ();
        ++file_number;
    } else
    if ( (strcmp (file_name,"-z",2) == 0) ||
        (strcmp (file_name,"-Z",2) == 0) ) {
        stat = zap_file ();
        ++file_number;
    } else
    if ( (strcmp (file_name,"-e",2) == 0) ||
        (strcmp (file_name,"-E",2) == 0) ) {
        stat = execute_file ();
        ++file_number;
    } else ++file_number;
    return stat;
}

```



```
get_xchar: get char from line
```

```
-----*/
int get_xchar ()
{
    int stat;
    int trys = 0;

    while ( ( (stat = asigetc (PORT)) < ASSUCCESS) && (trys < 10000) ){
        ++trys;
    }
    if (trys < 10000) return stat;
    return 0;
}
```

```
/*-----
send_xchar : send a char down line
```

```
-----*/
send_xchar (char c)
{
    int stat;
    while ( (stat = asiputc (PORT,c)) < ASSUCCESS) ;
}
```

```
/*-----
recieve_command
```

```
-----*/
char recieve_command (char c)
{
    int stat;
    int trys = 0;

    while ( ( (stat = get_xchar ()) != c) && (trys < 50) ) {
        ++ trys;
        /* send_xchar (NAK_CHAR); */
    }
    if (stat == c ) {
        send_xchar (ACK_CHAR);
        return TRUE;
    }
    return FALSE; /* error */
}
```

```
/*-----
send_command
```

```
-----*/
char send_command (char c)
{
    int stat;
    int trys;

    trys = 0;
    stat = asiputc (PORT,c);
    do {
        stat = get_xchar ();
        ++trys;
        if (stat == NAK_CHAR){
            send_xchar (c);
        }
    } while ( (stat != ACK_CHAR) && (trys < 100) );

    if (stat == ACK_CHAR)
        return TRUE;
    return FALSE;
}
```



```

/*-----
send_data: Basic function to send data
            format: 0F, #bytes follow, Bytes + XOR CRC
            trys 3 times then fails
-----*/
char send_data (char *s, int run) /* run, = len or size of data block */
{
    char bytes;
    int i, j, k, stat, return_value;

    k = 3;
    while (k) {
        timer (TICKS_PER_SECOND); /* wait for site to get in recieve mode */
        clrscr ();
        cprintf ("-* Sending Data...");
        bytes = run + 2; /* string + LRC + # bytes */
        j = run;
        send_xchar (bytes);
        for (i=0; i<j; i++)
            send_xchar(s[i]);
        send_xchar (calc_CRC (s, run));
        do {
            stat = get_xchar ();
        } while ( (stat != NAK_CHAR) && (stat != ACK_CHAR) );

        if (stat == NAK_CHAR) { /* Presumably a NAK char */
            --k;
            return_value = FALSE;
            clrscr ();
            cprintf ("-* Trying Send Data Again...");
            timer (TICKS_PER_SECOND);
        }
        if (stat == ACK_CHAR) {
            k = 0;
            return_value = TRUE;
        }
    }
    return return_value;
}

```

```

/*-----
calc_CRC
-----*/
char calc_CRC (char *s, int len)
{
    int i, j;
    char crc;

    crc = 0;
    i = len;
    crc = s[0];
    for (j=1; j<i; j++)
        crc = crc ^ s[j];
    return crc;
}

```

```

/*-----
// Function Name -> is_zero_file_length
// Parameters:

```



```
// Function:
// Returns:
// Written By : Greg McGregor
//
```

```
-----*/
int is_zero_file_length (char *file_name) {
int fd;
    fd = open (file_name,O_RDONLY|O_BINARY,S_IREAD);
    if (fd < 0 ) return ( FALSE );
    if ( filelength (fd) == 0L) { close ( fd ); return ( TRUE ); }
    close (fd);
    return ( FALSE );
}
```

```
/*-----
send_file
-----*/
```

```
int send_file ()
{
int stat;
    clrscr ();
    shift_left (file_name,2);
    cprintf ("-* Sending File '%s'",file_name);
    if (!send_command (0x01) )
        return FALSE;
    gotoxy (1,2);
    cprintf ("-* Got The OK!");
    timer (TICKS_PER_SECOND *2 );
    stat = YmodemSend (PORT,file_name,status_routine,NULL,ESC);
    if (is_zero_file_length (file_name) ) return ( TRUE );
    if (stat != XFER_RETURN_SUCCESS) return ( FALSE );
    return ( TRUE );
    gotoxy (1,3);
    cprintf ("-* File Transfer Status %d",stat);
}
```

```
/*-----
shift_left
-----*/
```

```
shift_left (char *s,int n)
{
int i;
    i = 0;
    while (s[i+n]) {
        s[i] = s[i+n];
        ++i;
    }
    s[i] = '\0';
}
```

```
/*-----
get_file
-----*/
```

```
int get_file ()
{
int stat;

    clrscr ();
    stat = 0;
    clrscr ();
    shift_left (file_name,2);
    cprintf ("-* Requesting File Transfer of '%s'",file_name);
    if (!send_command (0x02) )
        return FALSE;
}
```



```

gotoxy (1,2);
cprintf ("-* Request OK'd");
if (!send_data (file_name,strlen (file_name))) { /* tell site a file name */
    clrscr ();
    cprintf ("-* Couldn't Get The File '%s'",file_name);
    return FALSE;
} else {
    cprintf ("... Initiating Ymodem");
    stat = 0;
    stat = YmodemReceive (PORT,status_routine,NULL,ESC);
    if (is_zero_file_length (file_name) ) return (TRUE);
    if (stat != XFER_RETURN_SUCCESS) return FALSE;
    return TRUE;
    gotoxy (1,3);
    cprintf ("Transfer Value %d",stat);
}
}

```

```

/*-----
zap_file
-----*/

```

```

int zap_file ()
{
    int stat;

    clrscr ();
    stat = 0;
    clrscr ();
    shift_left (file_name,2);
    cprintf ("-* Requesting A File ZAP of '%s'",file_name);
    if (!send_command (0x0F) )
        return FALSE;
    gotoxy (1,2);
    cprintf ("-* Request OK'd");
    if (!send_data (file_name,strlen (file_name))) { /* tell site a file name */
        clrscr ();
        cprintf ("-* Couldn't ZAP The File '%s'",file_name);
    }
    hold_line (1); /* hold line for 1 second before continuing */
    return TRUE;
}

```

```

/*-----
execute_file
-----*/

```

```

int execute_file ()
{
    int stat;

    clrscr ();
    stat = 0;
    clrscr ();
    shift_left (file_name,2);
    cprintf ("-* Requesting A File EXECUTE of '%s'",file_name);
    if (!send_command (0x10) )
        return FALSE;
    gotoxy (1,2);
    cprintf ("-* Request OK'd");
    if (!send_data (file_name,strlen (file_name))) { /* tell site a file name */
        clrscr ();
        cprintf ("-* Couldn't EXECUTE The File '%s'",file_name);
    }
    hold_line (5);
    return TRUE;
}

```


}

void status_routine (char *m)

```
{
    gotoxy (1,3);
    cprintf ("                ");
    gotoxy (1,3);
    cprintf ("%s\n",m);
}
```

void transfer_status (XFER *b)

```
{
    gotoxy (1,5);
    cprintf ("Block Number : %ld",b->block_number);
    gotoxy (1,6);
    cprintf ("Byte Count   : %ld",b->byte_count);
}
```

/*-----

date_check () : set date and time at site

-----*/

date_check ()

```
{
    int stat1,stat;
    struct time t;
    struct date d;
    int size;

    size = (int) sizeof (t);
    stat = send_command (0x05); /* request a date set check */
    gettime (&t);
    stat = send_data (&t,size);
    size = (int) sizeof (d);
    getdate (&d);
    stat1 = send_data (&d,size);
    gotoxy (1,3);
    if ( (stat) && (stat1) ){
        cprintf ("-* Date/Time Set OK!");
    } else cprintf ("-* Date/Time NOT SET OK!");
    timer (TICKS_PER_SECOND*2); /* wait for it to get in command loop */
    return TRUE;
}
```

/*-----

archive_database :

-----*/

archive_database ()

```
{
    int stat;
    clrscr ();
    cprintf ("-* Requesting An Archive");
    stat = send_command (0x03); /* archive command */
    if (stat) {
        gotoxy (1,2);
        cprintf ("-* Archive OK'd");
    } else {
        gotoxy (1,2);
        cprintf ("-* Archive FAILED!");
        return FALSE;
    }
    hold_line (5); /* hold line 5 secs before continuing */
    return TRUE;
}
```



```

/*-----
dial_site
-----*/
dial_site ()
{
    int secs;
    char temp[100];
    char ch;

    clrscr ();
    cprintf ("-* Dialing %s",PHONE_NUMBER);
    HMSetDialingMethod (PORT, TOUCH_TONE);
    HMSetCarrier (PORT,ON);
    HMDial (PORT,PHONE_NUMBER);
    secs = 45;
    ch = ' ';
    while ( (secs > 0) && (!connected ()) && (ch != 0x1B) ){ /* ESC */
        gotoxy (1,3);
        cprintf ("Elapsed Time : %d ",(45 - secs));
        timer (TICKS_PER_SECOND);
        --secs;
        if (kbhit () ) ch = getch ();
    }
    if (connected ()) {
        gotoxy (1,5);
        cprintf ("-* CONNECTED!");
        start_commands ();
    } else {
        gotoxy (1,5);
        cprintf ("-* Couldn't Connect");
        hang_up ();
    }
}

/*-----
connect_to_site :  set up port start call transfer etc...
-----*/
connect_to_site ()
{
    windef main_win = {2,2,78,19,White,Blue,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
                        White,Blue};
    wintype main_wt;
    windef help_win  = {14,4,66,6,Yellow,Cyan,FALSE,FALSE,FALSE,TRUE,SINGLEFRAME,
                        Yellow,Cyan};
    wintype help_wt;

    ruff_area (1,1,80,23,White,Blue);
    main_wt = windowopen (&main_win);
    setttitle (main_wt,"*  GVN Network  *",CenterUpperTitle);
    help_wt = windowopen (&help_win);
    setttitle (help_wt,"* Note *",CenterUpperTitle);
    textcolor (Yellow + Blink);
    cprintf ("          Please wait until CIP is finished");
    status_wt = windowopen (&status_win);
    setttitle (status_wt,"CIP Transfer Utility V1.56",CenterUpperTitle);
    open_port (); /* port opened ok if made it to here */
    init_modem ();
    dial_site ();
    hang_up ();
}

```